Implementierung und Evaluierung eines beidseitigen Handovers am Robotersystem EDAN

Bachelorarbeit

für die Prüfung zum

Bachelor of Science

des Studiengangs Informationstechnik
an der Dualen Hochschule Baden-Württemberg Mannheim

von

Hanna Rebecca Riesch

26. August 2022

Bearbeitungszeitraum:	07.06.2022 bis 30.08.2022
Martikelnummer, Kurs:	2352321, TINF19IT1
Ausbildungsbetrieb:	Deutsches Zentrum für Luft- und Raumfahrt e.V.
Betrieblicher Betreuer:	Dipl.-Ing Jörn Vogel
Hochschulbetreuer:	Jürgen Schultheis

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema:

“Implementierung und Evaluierung eines beidseitigen Handovers am Robotersystem EDAN”

selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Hanna Rebecca Riesch
Oberpfaffenhofen, den 26. August 2022

Zusammenfassung

Diese Bachelorarbeit thematisiert ein beidseitiges Handover am Robotersystem EDAN. **E**MG-controlled **D**aily **A**ssista**N**t (EDAN) ist ein Assistenzroboter bestehend aus einem Rollstuhl und einem Roboterarm an dessen rechter Seite. Für die Umsetzung des Handovers wird als erstes eine Methode zur Bestimmung von 3D Gelenkpositionen ausgewählt und am System integriert.

Daraufhin werden die beiden Handoverrichtungen, vom Menschen zum Roboter und vom Roboter zum Menschen, mithilfe von **S**hared **C**ontrol **T**emplates (SCT) definiert.

Das Besondere an dieser Umsetzung ist, dass ein Benutzer den Roboter steuert. Ziel ist es, mittels der SCT die Übergabe für den Benutzer möglichst angenehm zu gestalten.

Für jede der beiden Handoverrichtungen werden drei Versionen erstellt. Diese werden abschließend mittels einer kleinen Nutzerstudie evaluiert.

Abstract

In this bachelor thesis, a two-sided handover between robot and human is implemented and evaluated. This is done on the robot system EDAN. **E**MG-controlled **D**aily **A**ssista**N**t is an assistance robot, containing a wheelchair and a robotic arm.

The first part of the thesis is the implementation of a method to detect the 3D position of selected human joints.

In the next step, three versions of the handover are implemented for both sides. For this purpose, the handover versions are defined with the use of Shared Control Templates (SCT).

The particular feature of the handover is the control of the robot system by a user. Therefore, the goal of this thesis is the user-friendliness during the handover.

A small user study is conducted to determine the user thoughts on the handover task. Based on the findings of the user study, the handover is evaluated.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt e. V.
EDAN	**E**MG-controlled **D**aily **A**ssista**N**t
R2H	**R**obot **to H**uman
H2R	**H**uman **to R**obot
SCT	**S**hared **C**ontrol **T**emplates
LN	**L**inks and **N**odes
RMC	**R**obotics and **M**echatronics **C**enter
COCO	**C**ommon **O**bjects in **CO**ntext
Cissy	**C**ontinuous **I**ntegration **S**oftware **SY**stem
WSR	**W**orld**s**tate **R**epresentaion
PDDL	**P**laning **D**omain **D**efinition **L**anguage
ODB	**O**bject **D**ata **B**ase
XML	E**X**tensible **M**arkup **L**anguage
GUI	**G**raphical **U**ser **I**nterface
IM	**I**nput **M**apping
AC	**A**ctive **C**onstraint
FII	**F**rame **I**nterpolator **I**nterface

ELAN	**E**MG-controlled **L**ab **A**ssista**N**t
CLASH	**C**ompilant **L**ow-cost **A**ntagonistic **S**ervo **H**and
HIT	**H**arbin **I**nstitute of **T**echnology
LWR	**L**ight **W**eight **R**obot
RGB	**R**ot **G**rün **B**lau
2D	zweidimensional
3D	dreidimensional
API	**A**pplication **P**rogramming **I**nterface
CNN	**C**onvolutional **N**eural **N**etwork
TCP	**T**ransmission **C**ontrol **P**rotocol
HMI	**H**uman **M**achine **I**nterface
EE	**E**nd**E**ffektor
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
TLX	**T**ask **L**oad **I**ndex
DHBW	**D**uale **H**ochschule **B**aden-**W**ürttemberg

1 Einleitung

In dieser Bachelorarbeit wird ein Konzept für die Implementierung und Evaluierung eines beidseitigen Handovers am Robotersystem **E**MG-controlled **D**aily **A**ssista**N**t (EDAN) entworfen und angewendet. In der Einleitung wird zuerst die Problemstellung der Arbeit erörtert, danach wird das Ziel der Ausarbeitung beschrieben. Im dritten Unterkapitel wird die Vorgehensweise zur Erstellung der Arbeit geschildert. Im letzten Schritt wird auf den Aufbau der Arbeit eingegangen.

1.1 Problemstellung

Zur Vorbereitung und um einen allgemeinen Überblick über den Stand der Forschung im Bereich der Robotikhandover zu erlangen, wurde in der T3_3000 Arbeit[1] eine ausführliche Recherche zu diesem Thema durchgeführt.

Eine Objektübergabe zwischen einem Roboter und einem Menschen wird als Handover bezeichnet. Dabei wird zwischen dem **R**obot **to H**uman (R2H) Handover und dem **H**uman **to R**obot (H2R) Handover unterschieden [2]. Beim R2H Handover wird der Gegenstand vom Roboter an den Menschen überreicht. Bei der anderen Möglichkeit, dem H2R Handover, wird das Objekt in der entgegengesetzten Richtung vom Menschen an den Roboter überreicht. Werden beide Handover ermöglicht, so entspricht es einem beidseitigen Handover. Beim Entwurf sowie bei der Implementierung können einige Herausforderungen auftreten. Die meisten hängen vom ausgewählten Robotersystem ab, für welches das Handover entworfen wird, da die Hard- und Softwarestruktur des Robotersystems die Komplexität des Handovers definiert.

1.2 Ziel der Arbeit

Das Ziel der Arbeit ist ein beidseitiges Handover für und am Robotersystem EDAN zu implementieren, integrieren und zu testen. Das Robotersystem EDAN ist ein Assistenzroboter, welcher es Menschen mit Einschränkungen ermöglicht, alltägliche Abläufe und Dinge, wie beispielsweise aus einem Glas trinken, wieder selbständig auszuführen.

1.3 Vorgehensweise

Im ersten Schritt soll das R2H Handover implementiert werden. Hierfür muss als erstes eine Methode gefunden werden, um die Position des Menschen zu bestimmen. Im Anschluss kann die Armbewegung des Roboters implementiert werden. Hierfür werden **S**hared **C**ontrol **T**emplates (SCT) angewendet. Der nächste Schritt ist es, das H2R zu implementieren. Dabei wird zuerst evaluiert, ob eine neue Objekterkennung für das Lokalisieren von stark verdeckten Objekten notwendig ist. Ist die bestehende Objekterkennung ausreichend für die Lokalisierung der Objekte in der Hand des Menschen, wird diese für das Handover verwendet.

Der letzte Schritt ist es eine Nutzerstudie / Pilotstudie durchzuführen, bei welcher das Handover bewertet wird.

1.4 Aufbau der Bachelorarbeit

Im ersten Schritt werden Aufgabenstellungen definiert. Diese sind in Kapitel 2 festgehalten. Danach werden die Methoden und Verfahren, die in dieser Bachelorarbeit angewendet werden, erläutert. Darunter fallen die verwendeten Programmiersprachen sowie zusätzliche Bibliotheken. In Kapitel 4, der Durchführung, wird das genaue Vorgehen zur Umsetzung der Aufgabenstellungen beschrieben. Im anschließenden Ergebniskapitel wird die Realisierung der Aufgabenstellungen bewertet. Das sechste und letzte Kapitel enthält die kritische Reflexion sowie den Ausblick.

2 Aufgabenstellungen

Im folgenden werden die Aufgabenstellungen dieser Bachelorarbeit aufgelistet.

1. Bestimmung der physischen Position des Menschen relativ zum Roboter und Verarbeitung dieser Position für das Handover.

   1.1 Literaturrecherche zur Positionsbestimmung des Menschen im Hinblick auf die Auswahl eines geeigneten Verfahrens für das Handover.

   1.2 Implementierung und Integration einer Methode zur Gelenkpositionsbestimmung des Menschen.

   1.3 Evaluierung der Möglichkeiten zur Bestimmung einer 3D Position des Menschen.

   1.4 Integration des Prozesses ins reale System sowie das Testen des Prozesses.

2. Implementieren und Testen der Übergabe von Gegenständen / Objekten von Roboter zum Menschen (Robot to Human R2H).

   2.1 Einarbeitung in Shared Control Templates (SCT).

   2.2 Recherche und Auswahl verschiedener Auslöser, welche für das Handover benötigt werden.

   2.3 Festlegung und Implementierung der Bewegungsabläufe, welche nach den Auslösern ausgeführt werden sollen.

   2.4 Einarbeitung in ein Modul zur Automatisierung von SCT, um das Handover zu automatisieren.

2.5 Integration des vollständigen Handovers am System sowie die Durchführung von Tests.

3. Integration einer neuen Objekterkennung, um verdeckte Objekte besser verfolgen zu können.

   3.1 Recherche und Einarbeitung in die Funktionsweise der neuen Objekterkennung.

   3.2 Implementierung eines Prozesses, welcher am Roboter angewendet werden kann.

   3.3 Integration des Prozesses sowie die Durchführung von Tests.

4. Implementierung und Testen des Human to Robot Handover (H2R).

   4.1 Recherche sowie Auswahl verschiedener Auslöser, welche für das Handover verwendet werden können.

   4.2 Festlegung und Implementierung der Bewegungsabläufe, welche nach den Auslösern ausgeführt werden sollen.

   4.3 Integration des vollständigen Handovers am System sowie die Durchführung von Tests.

5. Durchführung und Evaluierung einer Nutzerstudie.

   5.1 Recherche zur Bestimmung von Kriterien für die Nutzerstudie.

   5.2 Durchführung der Nutzerstudie und Anwendung der oben genannten Kriterien.

   5.3 Auswertung und Evaluierung der Nutzerstudie.

3 Methoden und Verfahren

In diesem Kapitel werden grundlegende Methoden und Verfahren erklärt, welche in allen Aufgaben Anwendung finden.

3.1 Programmiersprache

Die Softwaremodule des Robotersystems sind in unterschiedlichen Programmiersprachen geschrieben. Echtzeit kritische Prozesse sind am Robotersystem in den Programmiersprachen C oder C++ geschrieben. High-Level Module hingegen sind in den meisten Fällen in Python geschrieben. Diese werden für das Handover benötigt, weshalb für dieses hauptsächlich die Programmiersprache Python verwendet wird.

3.1.1 Python

Python ist eine objektorientierte Open Source Programmiersprache [3]. Einer der großen Vorteile von Python ist es, dass die Variablen während der Laufzeit zugewiesen werden. Für den Programmierer bedeutet dies, dass die Variablen nicht deklariert werden müssen, was ein Wechseln des Datentypes im Vergleich zu anderen Sprachen vereinfacht [4].

3.2 Bibliotheken

Für die verschiedenen Softwarekomponenten wurden unterschiedliche Bibliotheken verwendet. Im Folgenden werden zuerst die allgemein bekannten und öffentlichen

Bibliotheken mit ihren Versionsnummern aufgelistet. Im nächsten Schritt wird auf interne Bibliotheken des **R**obotics and **M**echatronics **C**enter (RMC) eingegangen, zu diesen erfolgt zusätzlich eine Erklärung.

3.2.1 Allgemein bekannte Bibliotheken und Softwarekomponeneten

Die folgenden öffentlich zugänglichen Bibliotheken werden für die Implementierung des Handovers verwendet:

Softwarekomponente	Versionsnummer	Verwendung
Python	2.7.16 und 3.6.12	verwendete Programmiersprache
Numpy	1.18.2	mathematische Berechnungen
OpenCV	4.2.0.32	Bildverarbeitungstools
math	in Python Version enthalten	mathematische Funktionen
sys	in Python Version enthalten	auslesen von Flags
time	in Python Version enthalten	stoppen der Zeit
torch	1.9.0	Deeplearning Framework
torchvision	0.9.0	Computer Vision Modul
pycocotools	2.0.2	zur Verarbeitung des **C**ommon **O**bjects in **CO**ntext (COCO) Datensatzes
lightweight-human-pose-estimation	-	Bestimmung der Gelenkpositionen in 2D

Tabelle 3.1: Öffentliche Softwarekomponenten

3.2.2 Interne Bibliotheken / Software

Die Tabelle 3.2 listet Software und Bibliotheken auf, welch institutsintern sind.

Softwarekomponente	Versionsnummer	Verwendung
SensorNet	1.1.2	Kamerafunktionalitäten
Cissy	1.7.1	Entwicklungspipeline
Links and Nodes	1.3	Prozesskommunikation
World State Representation	3.0.0	Welt Modell

Tabelle 3.2: Interne Softwarekomponenten

SensorNet

SensorNet stellt verschiedene Funktionalitäten zur Steuerung verschiedener Sensoren zur Verfügung. Ein möglicher Sensor ist die für diese Arbeit verwendete Asus Xtion Kamera, welche in Kapitel 3.5.1 beschrieben wird. Im Paket ist neben dem für die Kamera benötigten Treiber das `pysn.stream` Modul enthalten. Dieses kann in Python importiert werden und ermöglicht es, den Kamerastream zu verarbeiten.

Links and Nodes (LN)

Links and **N**odes (LN) wurde im RMC des **D**eutschen Zentrums für **L**uft- und **R**aumfahrt e.V. DLR entwickelt. Die Software ermöglicht es, verschiedene Prozesse auf verteilten Rechnerknoten zu verwalten. Zusätzlich stellt LN verschiedene Möglichkeiten für die Interprozesskommunikation bereit. Darunter zählen die Möglichkeiten zum Erstellen von kontinuierlicher sowie eventbasierter Kommunikation. Kontinuierliche Kommunikation wird mit einem Topic realisiert. Eventbasierte Kommunikation mit einem Service.

Topic Die Kommunikation über ein Topic ist ein Publisher-/Subscribermodell. Dies entspricht einer synchronen Kommunikation. Es gibt einen Prozess, welcher das Topic veröffentlicht. Idealerweise wird das Topic vom Publisher in einer festen Zeitrate veröffentlicht. Für jedes Topic kann es nur einen Publisher geben. Die Daten können von mehreren Subscriberprozessen gelesen werden.

Service Die Kommunikation über einen Service in LN entspricht dem Provider-/Clientmodell. Ein einzelner Prozess stellt den Service bereit, er ist der Provider. Zu diesem Service Provider können sich dann verschiedene Clients verbinden

und Anfragen versenden. Der Provider antwortet dem jeweiligen Client daraufhin. Dies entspricht einer asynchrone Kommunikation.

Welche Daten versendet werden können, wird in einer `Message Definition` definiert. Darin wird jedem Datum ein Datentyp und ein Name zugewiesen. Es wird für jeden Service und jedes veröffentlichte Topic eine `Message Definition` benötigt.

Cissy

Continuous **I**ntegration **S**oftware **SY**stem (Cissy) ist eine Entwicklungspipeline des RMC. Diese Pipeline vereint verschiedene Softwareengineeringtools. Es ist eine Kombination aus Conan, Artifactory, Jenkins sowie GitHub. Cissy wird verwendet, um verschiedenen Softwarepakete zu verwalten. Dies wird im RMC benötigt, da verschiedenen Softwarepakete auf unterschiedlichen Robotersystemen angewendet werden und diese unterschiedliche Betriebssysteme und Systemaufbauten verwenden. Durch die Kombination von LN und Cissy können unterschiedliche Prozessdefinitionen mit unterschiedlichen Abhängigkeiten und Softwaredefinition verwendet werden.

3.3 Softwaremodule und Software Architektur des Robotersystems

Im Folgenden werden verschiedene, für das Handover benötigte, Softwarekomponenten vorgestellt. Diese sind teilweise institutsintern. Sie werden an mehreren unabhängigen Robotersystemen des RMC angewendet.

Die Graphik 3.1 zeigt die verschiedenen Softwarekomponenten des Robotersystems.

Die Software des Robotersystems kann in drei verschiedene Bereiche aufgeteilt werden. In das User Interface, die Real-Time Prozesse und die für diese Arbeit relevante Shared Control Unit. Diese drei Komponenten werden zentral von einer High Level State Machine koordiniert. In den folgenden Unterkapitel werden die für dies Arbeit relevanten Module erklärt.

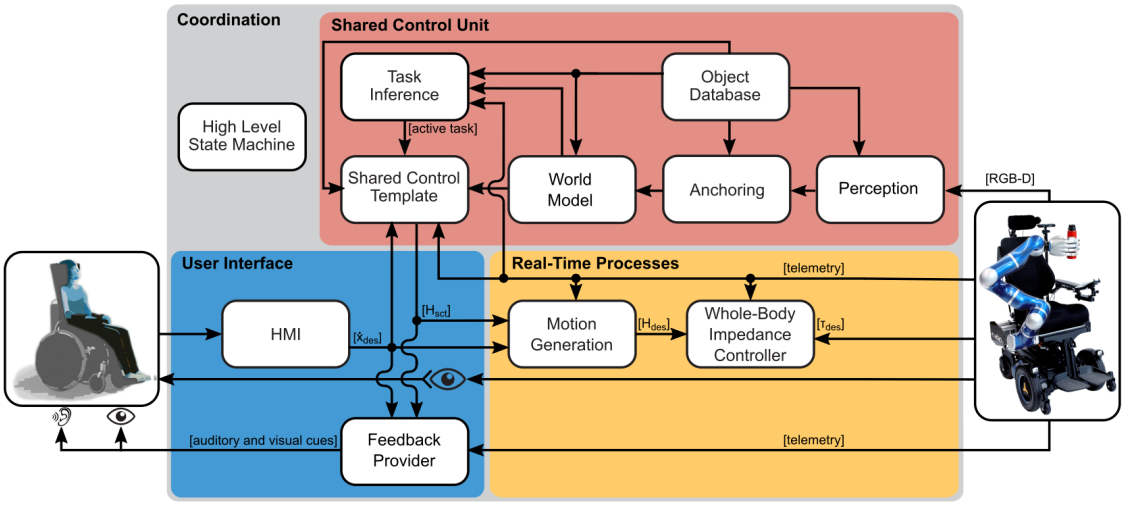


Abbildung 3.1: Die verschiedenen Software Module des Robotersystem EDAN. Basierend auf Abbildung 5 aus dem Paper "EDAN: An EMG-controlled Daily Assistant to Help People With Physical Disabilities" [5].

3.3.1 Object Database

Die **O**bject **D**ata **B**ase (ODB) ist eine Datenbank, welche alle Objekte, die für den Roboter verfügbar sind, verwaltet. Für jedes Objekt in der Datenbank muss ein Ordner erstellt werden. Dieser muss eine Manifest Datei enthalten, welche in der E**X**tensible **M**arkup **L**anguage (XML) Sprache geschrieben ist. Sie definiert in einer vorgegeben Struktur die grundsätzlichen Merkmale eines Objektes. Es ist möglich, dem Ordner des Objektes weitere Unterorder hinzuzufügen. Diese können Informationen für das Greifen, die Logik, die Geometrie oder die Darstellung enthalten. Die Abbildung 3.2 zeigt exemplarisch den Aufbau der ODB.

Im Logikordner werden die SCT Skills abgespeichert. Der Geometrieordner beschreibt die physikalischen Proportionen des Objektes. Im Darstellungsordner wird das Objekt gespeichert, wie es in der Visualisierung des Welt Modelles dargestellt wird. Der Graspordner enthält bestimmte Positionen am Objekt, an welchen dieses gegriffen werden kann.

Zusammenfassend werden in den Unterordnern Dateien gespeichert, welche auf das Objekt bezogen sind. Eine zusätzliche Möglichkeit ist es, einen Ordner beginnend mit einem Unterstrich zu definieren. Dieser wird verwendet, um Objektklassen zu

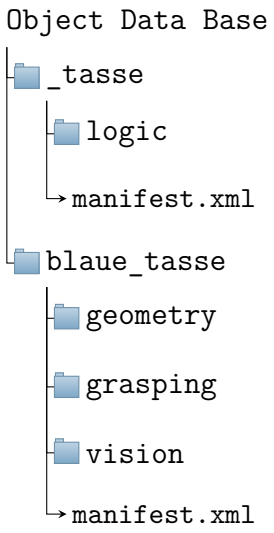


Abbildung 3.2: Ordnerstruktur der ODB

definieren. Einzelne Objekte können aus ihnen abgeleitet werden. Beispielsweise existiert die Überkategorie `_tasse`, zu dieser zählen dann alle Tassen, die in der ODB gespeichert sind. Diese Objekte können für die gleichen Aufgaben, welche im Logikordner der Klasse abgespeichert sind, verwendet werden.

Die in der ODB enthaltenen Daten werden von der Task Inference, dem Welt Modell und der Perzeption verwendet. Dies ist in der Graphik 3.1 veranschaulicht.

3.3.2 Anchoring

Das Anchoringmodul ist für die Verwaltung der Objekte zuständig. Daten aus der Perzeption werden an das Anchoring gesendet. Dieses stellt fest, ob ein Objekt bereits im vorherigen Iterationszyklus erkannt wurde und nimmt basierend darauf die Vergabe einer Identifikationsnummer für dieses Objektes vor. Zudem aktualisiert das Anchoring den Weltzustand des Welt Modells, wenn ein bestimmter Auslöser (**G**raphical **U**ser **I**nterface (GUI) Button oder Hardware Knopf) gedrückt wird.

3.3.3 Das Welt Modell

Die **W**orld**s**tate **R**epresentaion (WSR) spiegelt den aktuellen Stand des Welt Modells des Roboters wider. Sie enthält alle Objekte, die der Roboter erkannt hat. Neben Objekttyp und Objektname wird die Position des Objektes gespeichert. Es ist zudem möglich, bestimmte Semantiken der Objekte (wie beispielsweise "empfangsbereit") abzuspeichern [6]. Die WSR verwendet die **P**laning **D**omain **D**efinition **L**anguage (PDDL) [7] zur Definition und Verarbeitung dieser Semantiken. Ein weiteres Beispiel einer solchen Semantik ist das Binden eines Objektes an ein anderes. Es wird damit definiert, dass nachdem ein Objekt vom Robotersystem gegriffen wurde, das Objekt an die Hand des Roboters gebunden ist. Zur Visualisierung des Weltzustandes wird die Software RViz verwendet.

Die Informationen zu den einzelnen Objekten erhält die WSR aus der Perzeption und der ODB des Roboters. Der Zustand des Welt Modells wird von der Task Inference sowie den Shared Control Templates ausgelesen. Visualisiert in der Graphik 3.1.

3.3.4 Die Task Inference

Die Task Inference kann anhand des Weltzustandes vorhersagen, welche Aufgabe(n) in diesem Weltzustand möglich sind und liefert dadurch dem Benutzer des Roboters die Möglichkeit, eine dieser Aufgaben auszuwählen. Eine weitere Möglichkeit der Task Inference ist es, eine Aufgabenausführung unter bestimmten Umständen direkt zu starten. Beispielsweise wenn der Abstand zu einem Gegenstand einen bestimmten Grenzwert unterschreitet.

Die Task Inference erhält ihre Informationen aus der ODB, WSR sowie den Real-Time Prozessen (Graphik 3.1). Die in der Task Inference ermittelte aktive Aufgabe wird an die SCT weitergegeben. Diese werden in Kapitel 4.2.1 erläutert.

3.4 Ablauf beim Starten einer Aufgabe

Aufgaben am Robotersystem EDAN werden immer von einem Benutzer gestartet. Hierfür werden SCTs verwendet. Zusammengefasst ermöglichen diese Templates es, Aufgaben selbstständig als Benutzer auszuführen und dabei in bestimmten Bewegungsabläufen unterstützt zu werden [8]. Ein Beispiel ist das Greifen einer Flasche. Anstatt die Bewegungen komplett selbst auszuführen, werden bei der SCT Variante verschiedene Stadien der Aufgabe unterschiedlich unterstützt. Dies geschieht durch **I**nput **M**apping (IM), welches die eingegebene Bewegung auf die ausgegebene Bewegung abbildet. So wird beispielsweise aus einer Rechtsbewegung des Benutzers eine Drehung um eine bestimmte Achse. Eine weitere Möglichkeit sind **A**ctive **C**onstraint (AC). Als Beispiel wird in einem bestimmten Zustand nur das gerade Zufahren auf das Objekt erlaubt. Es gibt verschieden Wege eine Aufgabenausführung am Robotersystem EDAN zu starten. Diese Möglichkeiten sind:

- Manuelles Starten der Aufgabe über einen Button in der GUI des Roboters
- Starten der Aufgabe mithilfe der Task Inference

3.5 Verwendete Hardware

Entwickelt und getestet wird das Handover für und am Robotersystem EDAN. Dieser wird im RMC des **D**eutsches Zentrum für **L**uft- und **R**aumfahrt e. V. (DLR) in Oberpfaffenhofen entwickelt. EDAN besteht aus einem handelsüblichen Rollstuhl an dessen rechter Seite sich ein **L**ight **W**eight **R**obot (LWR)-Arm befindet. Das Robotersystem **E**MG-controlled **L**ab **A**ssista**N**t (ELAN) besitzt fast dieselben System- und Softwarekomponenten wie das Robotersystem EDAN. Der Hauptunterschied besteht darin, dass bei ELAN der LWR-Arm anstatt an einem Rollstuhl auf einer Plattform montiert ist. Da die Systemzeit am Robotersystem EDAN begrenzt ist, können Tests und Integrationen an diesem System vorgenommen werden. EDAN besitzt eine CLASH-Hand, während ELAN eine DLR-HIT-Hand besitzt. Die Softwareinfrastruktur an beiden Robotern ist identisch. Relevant für das Testen des Handovers ist, dass am Robotersystem ELAN keine Whole-Body Control durchgeführt werden kann. Whole-Body Control wird für Aufgaben verwendet, bei welchen

sowohl der Roboterarm als auch die Rollstuhlbewegung für das erfolgreiche Ausführen der Aufgabe benötigt werden [9]. In den folgenden Unterkapiteln wird auf Teile der Hardware des Robotersystems eingegangen. Dabei wird der Fokus auf die für diese Bachelorarbeit relevanten Systemkomponenten gelegt.

(a) Das Robotersystem EDAN

(b) Das Robotersystem ELAN

Abbildung 3.3: Bilder der beiden Robotersysteme.

Die Bilder 3.3 zeigen die beiden Roboter. Das Bild (a) zeigt EDAN, (b) zeigt ELAN.

3.5.1 Kameras

Um seine Umgebung wahrzunehmen, besitzt das Robotersystem verschiedene Kameras. Im Folgenden wird die Kamera erklärt, welche für die Objekterkennung sowie die Erkennung des Menschen verwendet wird. Neben diesen gibt es eine weitere Kamera, welche für die Teleoperation des Roboters verwendet wird. Diese hat für das Handover jedoch keine Relevanz, weshalb auf diese nicht genauer eingegangen wird.

Am Robotersystem wird die Asus Xtion Kamera verwendet. Diese sendet ein Infrarotmuster, welches mithilfe eines Infrarotsensors aufgenommen wird. Daraus wird intern der Tiefenwert generiert [10]. Dieses Verfahren ordnet jedem Pixel des **R**ot **G**rün **B**lau (RGB) Bildes einen Tiefenwert zu.

3.5.2 Roboterarm

EDAN und ELAN besitzen beide einen LWR3-Arm. Dieser wurde vom DLR speziell für die gemeinsame Arbeit mit Menschen entwickelt [11]. Der Arm besitzt mehrere kinematische Redundanzen, Drehmomentsensoren und sieben Freiheitsgrade [11]. Die eingebauten Kraftsensoren spielen eine wichtige Rolle für die Umsetzung des Handovers. Auf den Fotos der Roboter 3.3 ist jeweils auch der LWR-Arm zu erkennen.

3.5.3 Roboterhände

Ein Unterscheidungspunkt zwischen den beiden Robotersystemen ELAN und EDAN ist die verwendete Roboterhand. ELAN verwendet die **D**eutsches Zentrum für **L**uft- und **R**aumfahrt e. V. - **H**arbin **I**nstitute of **T**echnology (DLR-HIT)-Hand und EDAN die **C**ompilant **L**ow-cost **A**ntagonistic **S**ervo **H**and (CLASH).

DLR-HIT Hand

Im Unterschied zur CLASH-Hand besitzt die DLR-HIT-Hand fünf gleiche Finger [12]. Die Hand besitzt an jedem Finger vier Gelenke, was durch die Verkopplung zweier Gelenke zu jeweils drei Freiheitsgraden führt [12]. Die Hand besitzt wie auch die CLASH mehrere Sensoren.

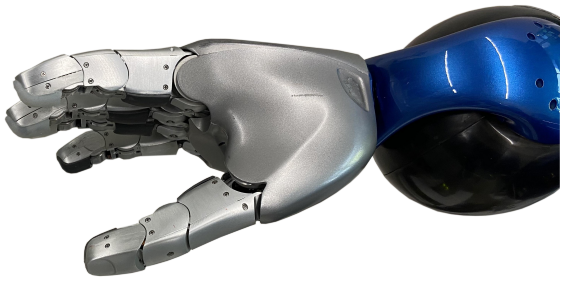


Abbildung 3.4: Die DLR-HIT-Hand.

Auf der Abbildung 3.4 ist ein Foto der DLR-HIT-Hand zu sehen.

Compilant Low-cost Antagonistic Servo Hand

Diese Roboterhand wurde für das Greifen von empfindlichen Lebensmitteln wie beispielsweise Obst entwickelt [13]. Die Seilverkopplung ist ähnlich der menschlichen Hand [14]. Besonders nützlich für die Anwendung im Handover Fall, aber auch in vielen anderen Bereichen, ist es, dass die CLASH viele verschiedene Sensoren besitzt. Neben acht Seilkraftsensoren enthält die Hand zu dem taktile Sensoren in der Handinnenfläche [14]. Ein weiteres Unterscheidungsmerkmal zu anderen Roboterhänden ist, dass die Finger mit unterschiedlicher mechanischer Steifigkeit angesteuert werden können. Dies ist besonders beim Greifen von empfindlichen Objekten sehr hilfreich [14]. Die Abbildung 3.5 zeigt ein Bild der CLASH

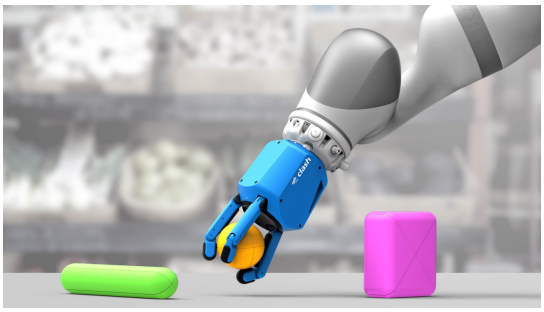


Abbildung 3.5: Die CLASH.[14]

4 Durchführung

In diesem Kapitel wird die Durchführung der Bachelorarbeit beschrieben. Dabei wird die Struktur der Aufgabenstellungen aus Kapitel 2 verwendet.

4.1 Bestimmung der physischen Position des Menschen und Verarbeitung dieser Position für das Handover

Mithilfe der physischen Position des Menschen ist dem System bekannt, an welchem Ort die Übergabe zwischen dem Roboter und dem Menschen stattfindet. Es gibt verschiedene Verfahren, um die Position zu bestimmen. Diese werden im folgenden ersten Unterkapitel vorgestellt und miteinander verglichen. Daraufhin folgt die Beschreibung der Integration des Kameratreibers sowie die Implementierung und Integration der Gelenkpositionserkennungssoftware. Nach deren Implementierung wird eine Evaluierung durchgeführt, welche Möglichkeiten bestehen, um aus den Daten der Gelenkpositionserkennungssoftware dreidimensional (3D) Koordinaten im Weltkoordinatensystem des Roboters zu generieren. Der letzte Schritt beinhaltet die Integration des Erkennungsprozesse im realen System sowie das Durchführen von Tests, ob die Erkennung im Robotersystemworkflow funktioniert.

4.1.1 Literaturrecherche zur Positionsbestimmung des Menschen im Hinblick auf die Auswahl eines geeigneten Verfahrens für das Handover

Das Wissen über die Position des Menschen spielt sowohl beim R2H Handover als auch beim H2R Handover eine wichtige Rolle. Im Folgenden werden drei verschiedene Softwaremöglichkeiten vorgestellt. Im Anschluss wird die Auswahl der Software begründet.

Softwaretools zur Bestimmung der Position des Menschen

Im Folgenden wird auf drei populäre Methoden zur Detektion eines Menschen in einem Bild / Video eingegangen. Es existieren viele weitere Möglichkeiten und Softwaretools. Die beiden im Folgenden zuerst erläuterten Methoden / Softwaretools werden ausgewählt, da diese bereits in Handoverszenarien in anderen Forschungseinrichtungen verwendet wurden [15][16]. Die dritte Methode wurde während der Recherche zu möglichen Alternativen gefunden.

Cascade Classifier von OpenCV OpenCV verfügt über Funktionen, welche es ermöglichen, das Gesicht eines Menschen, oder dessen gesamten Körper mit einer Boundingbox zu tracken(verfolgen). Das Gesicht / der Körper wird dabei mit der Haar cascade Methode detektiert. Die Funktion gibt dabei lediglich die vier Eckpunkte der Boundingbox zurück.

Diese Methode beruht auf dem Paper: *"Rapid Object Detection using a Boosted Cascade of Simple Features [17]"*. Dabei wird ein neuronales Netz erstellt. Dies wird mit vielen positiven und negativen Bildern trainiert [18]. Bilder, auf welchen Gesichter / Menschen zu erkennen sind, sind die positiven Bilder[18]. Auf negativen Bildern ist kein Gesicht / Mensch abgebildet [18].

Um die von OpenCV bereitgestellte Pythonfunktion zur Erkennung der Gesichter verwenden zu können, muss im Skript das Bild vorverarbeitet werden, da die Funktion mit schwarz-weiß Bildern arbeitet. Dies ist realisierbar mit weiteren, intern bereitgestellten, Funktionen von OpenCV.

Diese Methode wird in anderen wissenschaftlichen Ansätzen zur Umsetzung eines Handovers verwendet. In diesem Beispiel wird ausgehend von der Kopfposition der Übergabeort etwas unterhalb (circa Körpermitte) bestimmt [15].

OpenPose Dieses Softwartool besitzt sowohl ein **A**pplication **P**rogramming **I**nterface (API) für Python als auch für C++. Es wird die Erkennung von insgesamt 135 verschiedenen Keypoints ermöglicht [19]. Diese Erkennung wird in vier Module unterteilt. Es gibt die Erkennung des gesamten Körpers, die Erkennung von ausführlichen Gesichts, Hand oder Fuß Keypoints [19].

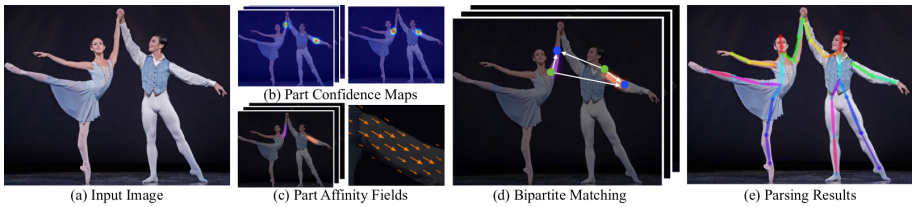


Abbildung 4.1: Verarbeitungsschritte von OpenPose [20]

Die Abbildung 4.1 zeigt die verwendete Pipeline von OpenPose. Im ersten Schritt (a) wird ein RGB Bild in ein **C**onvolutional **N**eural **N**etwork (CNN) gegeben[20]. Daraus wir zum einen eine Vertrauenskarte (b) für die Erkennung der Körperteile und Teilaffinitätsfelder (c) für die Teilverbindungen vorhergesagt[20]. Durch bipartiale Zuordnung (d) werden mögliche Körperteile gefunden[20]. Diese werden zu kompletten Gelenkpositionen eines Skelettes zusammengefasst[20].

LightweightOpenPose Diese Methode basiert auf der im vorherigen Kapitel erläuterten Funktionsweise von OpenPose. In der Anwendung ist der Hauptunterschied, dass lediglich 18 Gelenke erkannt werden können [21]. Dieses Modul besitzt keine Möglichkeit, zusätzliche Gelenkpunkte im Gesicht, den Händen oder den Füßen zu erkennen. Im Vergleich zu OpenPose wurde die Leistungsfähigkeit erhöht, dies führt zu einer schnelleren Performance [22]. Zudem ist die Methode einfacher zu installieren, da weniger zusätzliche Packages für die Installation benötigt werden. Es können direkt die fertigen Checkpoints des trainierten Neuronalen Netzes heruntergeladen werden[21].

Auswahl der Software

Für die Implementierung des Handovers am Robotersystem EDAN wird sich gegen die erste Methode entschieden, da kein exakt auf dem Menschen liegender Punkt bestimmt werden kann. Dies ist erforderlich, um die 3D Position zu berechnen. Die OpenCV Funktion gibt lediglich die Punkte der Boundingbox des Gesichtes oder des Körpers zurück. Diese Information wird als zweidimensional (2D) Pixelkoordinaten ausgegeben. Um diese in 3D Weltkoordinaten umzurechnen, muss an einem auf dem Menschen liegenden Punkt der Tiefenwert bestimmt werden. Wenn nur eine Boundingbox vorhanden ist, muss ein Workaround geschaffen werden. Ein weiterer Nachteil dieser Methode ist, dass keine Gelenke erkannt werden. Die Erkennung von Gelenken des Menschen kann für Erweiterungen verwendet werden. Eine mögliche Erweiterung ist, dass anhand der Verhältnisse zwischen den Gelenkpositionen bestimmt werden kann, ob der Mensch bereit ist für das Handover und dieses durchführen möchte.

Es wird sich für die Lightweight Variante entschieden, da diese gegenüber OpenPose den Vorteil der Performance besitzt und die Funktionalitäten für die Anwendung im Handover ausreichend sind.

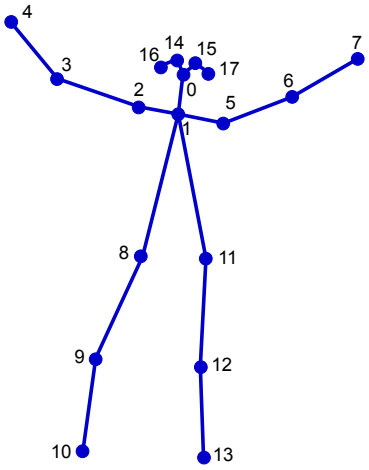


Abbildung 4.2: Darstellung der erkannten Gelenkpositionen von Lightweight OpenPose

Das Bild 4.2 zeigt die Keypoints, welche mit der Software erkannt werden können. Für das Handover spielen die Punkte 4 sowie 7 eine wichtige Rolle, da sich an diesen Stellen die Hände des Menschen befindet.

4.1.2 Implementierung und Integration einer Methode zur Gelenkpositionsbestimmung des Menschen

Im Folgenden wird beschrieben, wie die Gelenkpositionserkennung und der Kameratreiber im Gesamtsystem integriert werden. Die Integration am realen System erfolgt über den Weg der Simulation, in welcher die einzelnen Prozesse im ersten Schritt integriert werden. Der Aufbau sowie die Prozessstruktur der Simulation ist sehr ähnlich zur Struktur am realen System.

Der Kameratreiber

Um die Gelenkpositionserkennungssoftware auf einem eigenen Kamerastream anzuwenden, muss dieser in einem Pythonskript zur Verfügung stehen. Dafür wird `pysn.stream` aus dem SensorNet Paket (Kapitel 3.2.2) in das Pythonskript importiert. Dies ermöglicht es, einen Stream Channel im Skript zu erzeugen. Für dessen Erstellung muss der Name der Kamera bekannt sein. Dieser wird beim Starten des Treibers mitgegeben. Der gestartete Treiber speichert daraufhin den Kamerstream in einem shared memory space, welcher den vorgegebenen Namen besitzt. Dem erstellten Stream Channel im Skript wird dieser Namen ebenfalls mitgeteilt. Dies ermöglicht es, dass dieser den shared memory space auslesen kann und im Python Skript in der Form von zwei Matrizen zur Verfügung stellt. Es ist jeweils eine Matrix für das RGB-Bild und eine für das Tiefenbild verfügbar. Das RGB Bild ist wichtig, um die Gelenke des Menschen zu erkennen. Das Tiefenbild wird im späteren Verlauf verwendet, um die 3D Position des Menschen zu berechnen.

Einarbeitung in die Lightweight Open Pose Variante

Lightweight OpenPose kann über ein GitHub Repository heruntergeladen werden. Um die Lightweight OpenPose Variante am Robotersystem zu verwenden, werden viele verschiedenen Abhängigkeiten zu unterschiedlichen Paketen, intern als auch extern, benötigt. Diese sind ebenfalls in den Tabellen der verwendeten Softwarekomponenten unter 3.1 und 3.2 zu finden. Zudem ist es wichtig, dass Python3 und nicht Python2 verwendet wird. Diese Voraussetzung kann gewährleistet werden durch die Verwendung eines LN-Manager. In diesem können für jeden Prozess durch die Verbindung mit Cissy alle Abhängigkeiten definiert werden.

Im Repository liegt neben den benötigten Softwarekomponenten eine Demodatei. Über diese folgt die Einarbeitung in das Softwaremodul, zudem baut das finale Skript auf diese Datei auf.

Um die Gelenke im Bild des Kamerastreams der Asus Xtion Kamera erkennen zu können, muss der Kamerstream dieser im Pythonskript eingelesen werden. Dies geschieht über den im vorherigen Kapitel beschriebenen Kameratreiber und das damit verbundene Streammodul. Der Name des shared memory space kann beim Starten des Prozesses als Argument mitgegeben werden.

Nach diesem Schritt liegen sowohl die Tiefen- als auch die RGB-Matrizen vor. Die Gelenkpositionserkennung kann bereits auf den RGB-Stream angewendet werden. Sie liefert für jeden Menschen alle Gelenkpositionen in Pixelkoordinaten. Da am Handover immer nur ein Mensch teilnehmen kann, wird das Skript dahingehend angepasst, sodass immer nur ein Mensch erkannt werden kann.

4.1.3 Evaluierung der Möglichkeiten zur Bestimmung einer 3D Position des Menschen und Finalisierung des Gelenkpositionserkennungsskriptes

Die Lightweight OpenPose Gelenkpositionserkennung gibt die Position der einzelnen Koordinaten in Pixelkoordinaten wieder. Die Gelenkpositionen werden jedoch für das Handover als 3D Weltkoordinate benötigt. Im Folgenden wird beschrieben, wie die 2D

Position in 3D umgewandelt werden kann und wie dies im finalen Skript umgesetzt wird. Des Weiteren wird beschrieben, wie die Position im Weltkoordinatensystem an die restlichen Softwarekomponenten gesendet wird. Abschließend wird der finale Stand des Skriptes erläutert.

Konvertierung der 2D Position in 3D

Um die 3D Position im Raum zu erhalten, müssen Berechnungen mit dem Tiefenwert der Szene sowie den intrinsischen Werten der Kamera vorgenommen werden. Die intrinsischen Werte einer Kamera werden mithilfe der sogenannten K-Matrix dargestellt [23].

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

mit

f_x = Brennweite in X-Richtung
f_y = Brennweite in Y-Richtung
c_x = Position des optischen Sensors in Pixelkoordinaten in X-Richtung
c_y = Position des optischen Sensors in Pixelkoordinaten in Y-Richtung

Mit den in der Formel 4.1 beschriebenen intrinsischen Parameter der Kamera kann das Lochkameramodell angewendet werden. Um mit diesem die 3D Position zu bestimmen, werden die folgenden Formeln 4.2 - 4.4 [24] angewendet als *depth_scale* wird der Wert 1 verwendet:

$$x = \frac{(u - cx) \cdot z}{fx} \tag{4.2}$$

$$y = \frac{(v - cy) \cdot z}{fy} \tag{4.3}$$

$$z = -\frac{d}{depth\_scale} \tag{4.4}$$

mit

$$\begin{aligned} d &= \text{Tiefenwert} \\ u &= \text{Pixelkoordinate x} \\ v &= \text{Pixelkoordinate y} \\ x &= \text{3D Koordinate x} \\ y &= \text{3D Koordinate y} \\ z &= \text{3D Koordinate z} \\ depth\_scale &= \text{Skalierungsfaktor} \end{aligned}$$

Durch die oben beschriebene Umformung liegen die Positionsdaten als 3D Koordinaten im Kamerakoordinatensystem vor. Um diese am Robotersystem verwenden zu können, müssen noch drei weitere Anpassungen vorgenommen werden.

Zum einen werden die drei Koordinaten in eine 4×4 Matrix überführt, um diese für Homogenetransformationen verfügbar zu machen. Diese Matrix enthält sowohl die Rotation als auch die Translation eines Objektes. Bei der Gelenkpositionsbestimmung ist nur eine Translation vorhanden, die Rotationsanteile sind nicht bekannt.

Im nächsten Schritt muss das damit entstandene Koordinatensystem gedreht werden, da diese Berechnung nach den Standards von OpenCV erfolgt ist. Bei diesem Standard zeigt die Z-Achse aus der Kamera hinaus. Die Transformation vom Weltursprung des EDAN Welt Modell wird nach dem Standard von OpenGl angegeben. Bei diesem zeigt die Z-Achse in die Kamera. In diesem Fall darf nicht nur die Z-Achse der Position des Gelenkes umgedreht werden, da somit ein ungültiges Koordinatensystem

entstehen würde, weil dieses nicht mehr rechts händisch ist. Um dies zu verhindern, muss das gesamte Koordinatensystem rotiert werden. Dies wird mit Hilfe einer Rotationsmatrix vorgenommen.

$$ {}^{K_{gl}}T_{K_{cv}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5} $$

$$ K_{gl} = \text{Kamera nach OpenGl Definiton} $$
$$ K_{cv} = \text{Kamera nach OpenCV Definition} $$

Es wird die unter 4.5 beschriebene Matrix verwendet. Die darin beschriebene Rotation entspricht einer Drehung um 180° in der X-Achse. Um die Rotation anzuwenden wird diese mit der Matrix der Position des Gelenkes multipliziert.

Danach wird die Position in das Weltkoordinatensystem überführt. Dies geschieht mithilfe der bekannten Transformationsmatrix, welche die Transformation vom Weltursprung zur Kamera widerspiegelt. Um diese Matrix zu erhalten, muss die Kamera zur Roboterbasis kalibriert werden. Nach dieser Kalibrierung kann die Matrix für die Transformation der 3D Gelenkposition in das Weltkoordinatensystem verwendet werden. Das Skript erhält diese Matrix über die LN Infrastruktur, dort wird die Transformation über die Benutzereinstellungen zur Verfügung gestellt. Diese werden in einem Topic veröffentlicht. Sie können nach der Integration des Prozesses in den LN Manager des Roboters vom Gelenkpositionserkennungsprozess gelesen werden. Durch die Anwendung der Formeln des Lochkameramodells und der Anwendung der Rotationsmatrix ist die Transformation ausgehend von der Kamera zum Gelenk bekannt. Diese kann im nächsten Schritt hin zum Koordinatenursprung des EDAN Welt Modells mit der folgende Formel 4.6 transformiert werden:

$$ {}^{W}T_{M} \;=\; {}^{W}T_{K_{gl}} \cdot {}^{K_{gl}}T_{K_{cv}} \cdot {}^{K_{cv}}T_{M} \tag{4.6} $$

mit

$$M = \text{Mensch}$$
$$W = \text{Weltursprung}$$
$$K_{gl} = \text{Kamera nach OpenGl Definiton}$$
$$K_{cv} = \text{Kamera nach OpenCV Definition}$$

Diese Formel resultiert aus der folgenden Graphik 4.3.

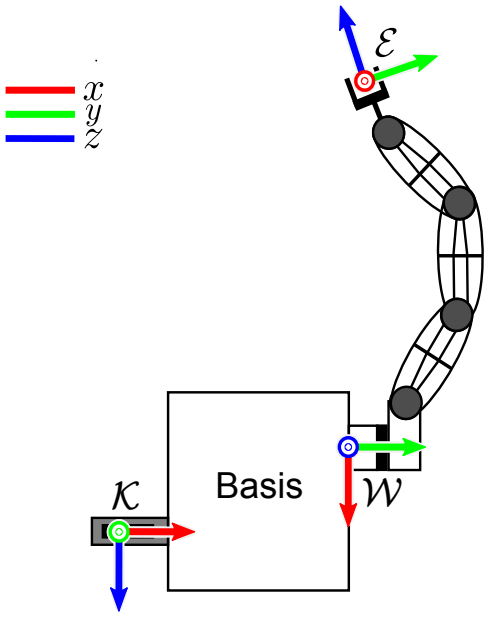


Abbildung 4.3: Koordinatensysteme an den Robotersystemen, welche für diese Bachelorarbeit benötigt werden. Basierend auf Abbildung 6 aus dem Paper “EDAN: An EMG-controlled Daily Assistant to Help People With Physical Disabilities”[5].

In der Graphik 4.3 sind die beiden in der Formel angewendeten Koordinatensysteme zu sehen. Die Koordinatensysteme sind sowohl am Robotersystem ELAN als auch am System EDAN an der linken und rechten Seite des Roboters. Die genaue Position ist unterschiedlich. Um die Matrix ${}^{W}T_{K}$ zu erhalten, wird eine Kalibrierungsprozess durchgeführt.

Nach dieser Transformation kann die daraus resultierende Position an weitere Softwarekomponenten des Robotersystems weitergegeben werden.

Senden der 3D Position an das EDAN Welt Modell

Wird ein Objekt zum Welt Modell hinzugefügt, geschieht dies durch das Anchoring. Dieses wurde für rein statische Objekte entwickelt, welche sich typischerweise nicht bewegen. Würde der Mensch über das Anchoring hinzugefügt werden, würde dies zu Problemen im Anchoring führen, da die Iterationen im Anchoring nicht schnell genug für die Bewegung des Menschen sind und zudem ein Durchschnittswert zwischen den Iterationen im Anchoring gebildet wird. Dies würde darin resultieren, dass mehrere Menschen währen einer Bewegung eines einzelnen Menschen erkannt werden würden. Dies kann sehr verwirrend für einen Benutzer des Systems sein.

Da das Anchoring weiterhin nur für statische Objekte verwendet werden soll, kommt eine Ausnahmebehandlung des Menschen im Anchoring nicht in Frage. Aus diesem Grund wird das Anchoring übersprungen und nicht verwendet.

Das Hinzufügen des Menschen wird im Gelenkerkennungsprozess realisiert. Da die WSR Funktionen nur in Python2 Skripten verwendet werden können, muss im Skript jede benötigte Funktionalität der WSR über eine spezielle Wrapperklasse importiert werden. Mit diesen importierten Funktionen kann der Mensch zur WSR hinzugefügt werden. Da nur ein einziger Mensch erkannt wird, ist es nicht notwendig, diesem eine Identifikationsnummer zuzuweisen. Wird ein Mensch erkannt, so wird dieser zu der WSR hinzugefügt, soweit dieser noch nicht bereits enthalten ist. Ist er bereits in der WSR, wird seine Position aktualisiert.

Für ein reibungsloses Funktionieren der SCT Skills darf das Objekt, welches für diese Aufgabe benötigt wird, nicht während der Aufgabenausführung aktualisiert werden. Aus diesem Grund wird vor dem Aktualisieren des Objektes überprüft, ob der Roboter im aktuellen Moment in einer Aufgabenausführung ist, in welcher der Mensch verwendet wird. Ist dies der Fall, wird der Mensch in der WSR nicht aktualisiert. Die hierfür benötigten Informationen erhält der Gelenkpositionsbestimmungsprozess über ein LN-Topic.

4.1.4 Finales Skript

Das finale Skript ist wie folgt aufgebaut:

Imports Zuerst werden alle benötigten Imports vorgenommen. Es werden allgemein bekannte Bibliotheken, Module des Lightweight OpenPose Projektes, Bibliotheken des RMC sowie spezielle Funktionen aus dem EDAN Projekt importiert.

Allgmeines Setup Zu Beginn werden alle LN-Clients und Subscriber deklariert.

Einlesen der Argumente Im nächsten Schritt werden alle Argumente, welche beim starten des Prozesses mitgegeben werden, ausgelesen. Diese sind:

- checkpoint-path: Dieses gibt an, an welcher Stelle die Checkpoints für das Neuronale Netz gespeichert sind.
- cpu: Dieses gibt an, ob die CPU verwendet werden soll.
- camera-name: Dieses gibt den Namen des shared memory space an, welcher verarbeitet werden soll.

Initales set up Darunter zählt das Einlesen der Checkpoints, das Initialisieren des Kamerastreams sowie das Einlesen der ${}^{W}T_{K}$ Matrix.

Detektionsschleife In dieser wird jedes Bild verarbeitet und die Position der Gelenke bestimmt. Zudem wird in dieser Schleife jeweils die Position an die WSR weitergegeben.

4.1.5 Integration des Prozesses ins reale System sowie das Testen am System

In diesem Unterkapitel wird als erstes auf die Visualisierung der Gelenkpositionen im System eingegangen. Daraufhin wird die Integration des Erkennungsprozess im System geschildert. Abschließend wird erläutert, wie die Funktionsweise des Prozesses getestet wird.

Visualisierung

Lightweight OpenPose besitzt bereits Funktionen, um die Gelenkpostionen in einem Videostream zu visualisieren. Diese Funktionen verwenden die `imshow` Funktion von OpenCV. Die bereits bestehende Visualisierung wird dahingehend erweitert, dass die Gelenkpunkte vergrößert werden und eine auffälligere Farbe für die Darstellung verwendet wird. Durch Angabe des auszugebenden Displays in der Prozessdefinition kann der Gelenkerkennungsprozess auf dem Rechner, auf welchem der Prozess läuft, den Videostream anzeigen. In diesem Video werden die Gelenkpunkte des Menschen dargestellt. Diese werden zu einem Skelett verbunden. Dadurch ergibt sich der folgende Output 4.4.

Abbildung 4.4: Beispielbild des Videostreams der Gelenkpositionserkennungssoftware

In der dreidimensionalen RViz Simulation des Roboters wird die Position der rechten Hand des Menschen hinzugefügt. Die Position wird mit einem Modell der CLASH markiert. Hierfür müssen Dateien, welche die Geometrie der CLASH enthalten, dem Unterordner `geometry` des `human` Objektes in der ODB hinzugefügt werden.

Integration

Für die Erkennung des Menschen (beziehungsweise seiner Hand) wird ein eigener Prozess im LN-Manager des Roboters erstellt. Im LN Manager wird der Prozess im Perzeptionsmodul abgelegt. Zu beachten ist, dass der Prozess auf einen Rechner gelegt wird, welcher über eine GPU verfügt. Am realen System muss hierfür der Kamerastream über mehrere Rechner geleitet werden. Die Graphik 4.5 zeigt den Datenfluss der Kameradaten im System.

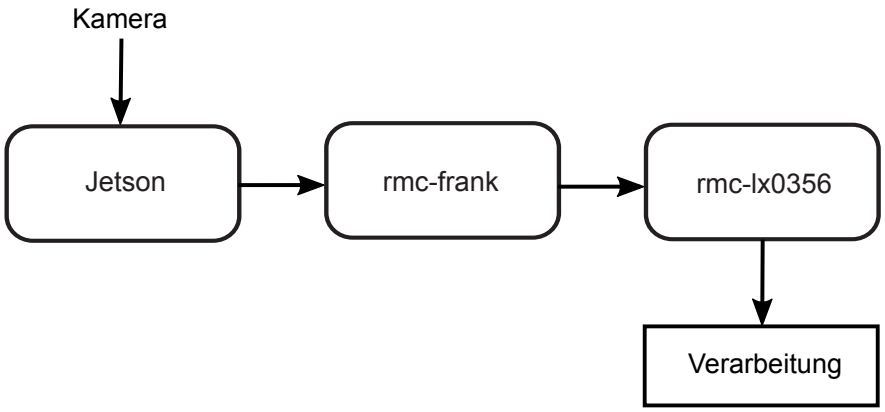


Abbildung 4.5: Datenfluss des Kamerastreams am realen System

Das SensorNet Paket legt für den Kamerastream einen shared memory space an, in welchem die jeweils aktuellen Kamerabilder abgelegt werden. Um diesen Ordner auf andere Rechner zu versenden, gibt es ein Skript im Paket. Mit Angabe der IP-Adresse sowie des Portes kann über zwei Prozesse der Stream weitergeleitet werden. Zum Weiterleiten wird ein Prozess für das Senden des Streams und ein weiterer Prozess zum Empfangen des Streams benötigt. Für die Übertragung über das Netzwerk wird das **T**ransmission **C**ontrol **P**rotocol (TCP) verwendet.

Testen

Zur Überprüfung wird zum einen die Simulation als auch das reale System verwendet. In der Simulation wird vom Gelenkpositionsbestimmungsprozess ein Objekt in das Welt Modell eingefügt. Die Simulation verfügt über verschiedene Werkzeuge. Eines dieser Werkzeuge ist ein virtueller Zollstock, mit welchem die Abstände von Objekten in der Simulation gemessen werden können. Eine weitere Möglichkeit zur Überprüfung ist es, sich die verschiedenen Matrizen ausgeben zu lassen und dann mittels eines Pythonskriptes den euklidischen Abstand dieser zu bestimmen. Mit diesen Methoden lässt sich die Genauigkeit der Gelenkpositionsbestimmung ermitteln. Zudem ist es möglich, die Position anstatt in Weltkoordinaten in Kamerakoordinaten ausgeben zu lassen. Mithilfe eines Maßbandes können diese Werte ebenfalls validiert werden. Hierfür werden mittels des Maßbandes Punkte im Raum mittels xyz Koordinaten gemessen. Der mit dem Maßband gemessene Wert wird mit den Werten des Gelenkpositionsbestimmungsprozesses verglichen.

Die Tests am realen System und in der Simulation laufen gleich ab. Während das reale System läuft, läuft parallel auch die Simulationssoftware mit.

Eine weitere Möglichkeit zu testen, ob die Erkennung präzise ist, ist es, wenn der Mensch einen Gegenstand in seiner Hand hält, welcher von der Perception des Roboters lokalisiert werden kann. In diesem Fall müssen die Objekte sehr nah beieinander hinzugefügt werden. Ist dies nicht der Fall, ist ein Offset in der Erkennung vorhanden.

Analyse des Prozesses

Der Gelenkpositionsbestimmungsprozess benötigt zwischen 0.1 und 0.15 Sekunden je Durchgang. Dies entspricht einer Aktualisierungsrate zwischen 6.6 und 10 Hertz. In den relevanten Bereichen für das Handover wird die Gelenkposition ausreichend gut lokalisiert, um dieses erfolgreich durchführen zu können. In den Bereichen außerhalb wird der Fehler in der Lokalisierung größer.

4.2 Implementieren und Testen der Übergabe von Gegenständen / Objekten vom Roboter zum Menschen (Robot to Human R2H)

In diesem Kapitel wird das Vorgehen zur Implementierung des Handovers R2H beschrieben. Zudem wird auf das Testen von diesem eingegangen.

4.2.1 Einarbeitung in Shared Control Templates (SCT)

Für die Ausführung des Handovers werden SCTs verwendet. Diese erlauben es, dass der Benutzer die Aufgaben zwar selbst steuert, dabei aber Unterstützung durch die zuvor definierten Skills erhält [8]. Ein Skill wird dabei im Format der SCT definiert. Ein Skill entspricht einem deterministischen Automaten. Im Folgenden wird zuerst auf die Grundlagen der SCTs eingegangen, daraufhin wird der grundsätzliche Aufbau eines solchen Templates erklärt.

Grundlagen der SCT

Im Folgenden werden zuerst grundlegende Begriffe für das Verständnis der SCT erklärt. Die SCT sind als Softwaremodul bereits am Robotersystem integriert und basieren auf dem Paper: “Shared Control Templates for Assistive Robotics” [8].

Input Mapping Das **I**nput **M**apping (IM) bildet die eingegebenen Werte xyz auf die Roboterbewegung ab. Die Werte können über ein **H**uman **M**achine **I**nterface (HMI) mit drei verschiedenen Signalen eingelesen werden. Dabei liegen die Werte x, y und z vor, diese beschreiben jeweils eine Eingabe in der zugehörigen Achse. Diese Eingabe wird auf ein sechsstelliges Array abgebildet. Dabei entsprechen die ersten drei Werte einer Translation entlang der jeweiligen Achsen. Die letzten drei entsprechen einer Rotation um diese Achse. Zur Veranschaulichung wird im Folgenden unter Listing 4.1 ein IM erläutert, welches verwendet wird, um sich im Handover nach der Übergabe des Objektes vom Menschen zu entfernen.

Listing 4.1: Input Mapping: Bewegung entlang einer Achse in nur einer Richtung

```
1  input_mapping:
2    - relative: True
3      mapping: [0,0,alpha-,0,0,0]
4      auxiliary_functions:
5       alpha:
6         function: Scalar
7         vector: {origin_frame: EE, axis: Z}
8         partial_input_mapping:
9          mapping: [tx, ty, 0, 0, 0, 0]
10   - mapping: [0,0,.3tz+,0,0,0]
```

Eingeleitet wird das IM in der 1. Zeile mit dem Signalwort `input_mapping`. Das IM ist in zwei Teile aufgeteilt, dies ist erkennbar anhand der beiden Aufzählungszeichen. Das erste IM ist in den Zeilen 2-9 definiert. In diesem wird die Hilfsfunktion `alpha` verwendet. Die Ergebnisse dieser Funktion werden auf eine Translation in der Z-Achse abgebildet. Dies wird in der Zeile 3 Definiert.

Das "`-`" bedeutet, es werden nur die negativen Werte von dieser Funktion verwendet. Durch die Verwendung von `relative: True` wird definiert, dass das Mapping im angegebenen Koordinatensystem verwendet werden soll, in diesem Fall ist dies das Koordinatensystem des **E**nd**E**ffektor (EE). Die Hilfsfunktion wird in den Zeilen 5-9 definiert. `Alpha` bildet das Skalarprodukt aus den Eingaben der Achsen X und Y. Zudem wird für das Skalarprodukt ein Vektor entlang der Z-Achse des Koordinatensystems des EE gebildet. Auf diesen wird das Skalarprodukt projiziert. An dieser Stelle (Zeile 7) wäre es zudem möglich, Vektoren zwischen zwei unterschiedlichen Koordinatensystemen zu definieren.

Das zweite IM ist in Zeile 10 definiert. Da `relativ` nicht gesetzt wird, bezieht sich dieses IM auf das Ursprungskoordinatensystem des EDAN Welt Modells. Es wird die Eingabe in der Z-Achse auf die Z-Achse im Welt Modell abgebildet. Die Geschwindigkeit dieser wird verlangsamt auf 30% der Eingabe.

Active Constraint Die **A**ctive **C**onstraints AC beschreiben geometrische Zwangsbedingungen, welche während eines Zustand des Automaten gelten. Diese können

bestimmte Werte widerspiegeln oder aus komplexeren Funktionen bestehen. Das Listing 4.2 ist ein Beispiel für AC, welches im Handover verwendet wird, um die Hand in die Richtung des Menschen zu orientieren.

Listing 4.2: Active Constraint: Orientierung der Hand in die Richtung des Targets

```
1  active_constraints:
2   - frame: EE
3     ref: {origin_frame: EE,
          orientation_to_frame: {origin_frame:
          target}}
4     mapping: [x,x,x,-0.4,x,x]
```

Das Schlüsselwort `active_constraints` leitet die Bedingungsdefiniton ein. In Zeile 3 wird ein Vektor ausgehend vom EE hin zum `target`, in diesem Fall der Mensch, definiert. Die Drehung in der X-Achse des EE wird in der Zeile 4 definiert. Diese wird benötigt, sodass die Handöffnung zum Menschen zeigt.

Um einen Skill zu definieren, muss dieser in eine YAML-Datei geschrieben werden. Diese Datei muss dann in der ODB einem Objekt zugeordnet werden. Durch das Abspeichern des Skills im Logikordner des Objektes, sowie das Hinzufügen des Dateinamens in der Manifestdatei, wird der Skill diesem Objekt hinzugefügt. Es ist möglich, den Skill in den Logikordner einer Klasse abzuspeichern, dadurch ist dieser für alle Unterkategorien verfügbar.

Aufbau eines SCT Skills

Die Skill-Datei lässt sich in zwei Bereiche einteilen. In den initialen Definitionsbereich und in den Bereich, in welchem der Automat definiert wird. In der Datei wird dabei mit dem Definitionsbereich begonnen. Daraufhin folgt die Definition des Automaten.

Initialer Definitionsbereich Dieser Bereich befindet sich direkt zu Beginn der Datei. Diese Definitionen lassen sich in zwei Kategorien gliedern. Zum einen gibt es Definitionen, welche für die Automatisierung des Skills benötigt werden.

Auf diese wird im späteren Verlauf der Bachelorarbeit unter Kapitel 4.2.4 eingegangen.

Zum anderen werden Definition in der PDDL Sprache vorgenommen. Diese geben Bedingungen an den Weltzustand an. Die Bedingungen müssen erfüllt sein, dass der Skill ausgeführt werden kann. Sind diese Bedingungen nicht erfüllt, so kann der Skill nicht gestartet werden. Das Listing 4.3 zeigt beispielhaft die initiale Definition eines R2H Handover Skills.

Listing 4.3: Beispiel einer initialen Definition zu Beginn eines SCT Skills

```
1 parameters: (?s - _container ?t - human
              ?m - _manipulator)
2 precondition: (and (bound ?s ?m)
                (receive_ready ?t))
3 actor: (?s - _container)
4 target: (?t - human)
```

Eine Definition, welche immer vorhanden sein muss, ist die `parameters` Definition. Diese gibt an, welche Objekte aus der ODB in der WSR vorhanden sein müssen, um die Aufgabe ausführen zu können. In Listing 4.3 wird definiert, dass im Welt Modell drei Objekt vorhanden sein müssen für diese Aufgabe. Ein Objekt der Klasse `container`, ein Objekt der Klasse `human` und ein Objekt der Klasse `manipulator` muss vorhanden sein. Dies wird in der ersten Zeile des Listings 4.3 definiert.

Mit der `precondition` Definition können den zuvor definierten Objekten zusätzliche Bedingungen auferlegt werden, um die Aufgabe starten zu können. In Listing 4.3 werden in der zweiten Zeile zwei Bedingungen aufgestellt, welche beide erfüllt sein müssen. Der `container` muss an den `manipulator` gebunden sein und der `human` in der Eigenschaft `receive_ready` (Empfangsbereit) den Wert `True` gesetzt haben.

Zudem wird in diesem Stil festgelegt, welches der unter `parameters` definierten Objekte der `manipulator` (Roboter) und welches das `target` (Ziel) der Aufgabe ist. Diese Informationen werden in der PDDL Sprache definiert, die darin verwendeten Bezeichnungen können bei der späteren Definition des Automaten

wiederverwendet werden. Das Listing 4.3 zeigt eine solche Zuordnung in den Zeilen 3 und 4.

Automaten Definition Nach den zuvor beschriebenen initialen Definitionen beginnt die Definition des Automaten mit dem Schlüsselwort `states`. Darunter werden die einzelnen Zustände des Automaten definiert. Jeder Zustand beginnt mit seinem Namen. Daraufhin ist es möglich in jedem Zustand unterschiedliche Definitionen vorzunehmen. Es muss nicht jede Einstellungsmöglichkeit in jedem Zustand verwendet werden. Die Definitionsmöglichkeiten werden im Folgenden aufgezählt:

parameters In diesem Bereich lassen sich unterschiedliche Parameter für diesen Zustand definieren. Es können Definitionen bezüglich des Verhaltens des Arms vorgenommen werden. In diesem Punkt kann zudem die Konfiguration für den EE in diesem Automatenzustand definiert werden. Damit kann beispielsweise festgelegt werden, dass sich in diesem Zustand die Hand öffnen soll.

transitions In diesem Abschnitt werden die Übergänge ausgehend vom aktuellen Zustand (in welchem diese Definition stattfindet) hin zu anderen Zuständen definiert. Jeder Übergang wird mit dem Schlüsselwort `to` eingeleitet, dann wird der Name des Zielzustandes angegeben. Es können verschiedene Bedingungen für den Übergang definiert werden. Sobald diese erfüllt sind, wird der Zustand gewechselt. Diese Bedingungen können komplexe Funktionen sein, welche die Positionen von Objekten oder auch die einwirkende Kraft auf den Roboterarm miteinbezieht. Eine andere Möglichkeit ist es, ein Timeout zu verwenden, nach welchem der Zustand gewechselt wird.

input mapping Die Funktionsweise wurde im vorherigen Kapitel 4.2.1 beschrieben.

active constraints Die Zuständigkeit und das Erstellen / Definieren der Constraints wurde in Kapitel 4.2.1 erläutert.

effect Dieser Effekt bezieht sich auf die PDDL. Damit kann der Zustand eines Objektes in der WSR auf einen bestimmten Wert gesetzt werden. Zum

Beispiel, dass die Hand des Roboters nach dem R2H Handover leer ist und das überreichte Objekt nicht mehr an den Roboterarm gebunden ist.

Listing 4.4: Beispiel einer initialen Definition zu Beginn eines SCT Skills

```
1 effect: (and (not (bound ?s ?m)) (free ?m))
```

Die Definition des zuvor beschriebenen Effektes ist in Listing 4.4 zu sehen.

4.2.2 Evaluierung und Auswahl verschiedener Auslöser und Vorbedingungen, welche für das Handover benötigt / verwendet werden

Für das R2H Handover werden mehrere Auslöser benötigt. Direkt zu Beginn wird ein Auslöser benötigt, welcher das Handover startet. Wenn dieses ausgeführt wird, werden für alle Zustände, die der Automat besitzt, Auslöser für den Übergang zwischen diesen benötigt. Diese Übergänge werden in der Definition des Automaten beschrieben.

Allgemeine Übersicht über mögliche Auslöser

Es gibt eine Großzahl an möglichen Auslösern für das Starten des Handovers [1]. Diese Auslöser lassen sich wie folgt gruppieren[1] :

- **Eingabesysteme** Diese können beispielsweise ein Knopf in einer GUI oder ein Hardwareknopf sein.
- **Sensoren** Es können verschiedene Sensoren verwendet werden, um Daten zu generieren, auf welchen die Entscheidung des Startens des Handovers basiert. Beispiele sind ein Mikrofonsensor oder ein Kamerasensor. Mithilfe des Mikrofons kann eine Stimme eines Menschens aufgenommen und analysiert werden. Bei der Erkennung eines bestimmten Schlüsselwortes wird das Handover gestartet[1]. Wenn der Kamerasensor verwendet wird, können dessen Bilder in eine Bildverarbeitung gegeben werden, in welcher dann die Umgebung des Roboters erkannt und verarbeitet wird [25]. Wird ein Mensch auf dem Bild

erkannt, so kann dessen Haltung und Gestik interpretiert werden. Dies kann als Auslöser für das Handover verwendet werden. Eine weitere Möglichkeit ist die Verwendung von Abstandssensoren oder Kraftsensoren. Anstatt eines Abstandsensors, kann der Abstand zu einem Objekt durch die Bildverarbeitung berechnet werden.

Auswahl der Auslöser für das Robotersystem

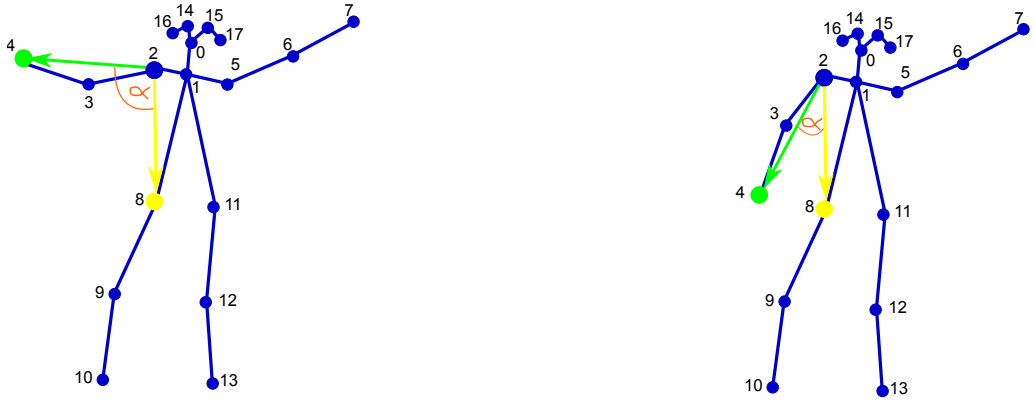
Nicht alle der im vorherigen Kapitel genannten Auslöser sind am Robotersystem EDAN vorhanden. Es folgt eine Auflistung, welche Auslöser in dem aktuellen Zustand des Roboters angewendet werden können.

Button Für das Testen am realen System sowie in der Simulation ist die Verwendung eines Buttons in einer GUI die einfachste Variante. Durch das Drücken des Buttons wird die Aufgabe direkt gestartet. Der Button kann in einer bereits existierenden GUI hinzugefügt werden. Dabei wird direkt definiert, welche Objekte für das Handover verwendet werden sollen. Zudem kann hinzugefügt werden, dass beim Drücken des Buttons nicht nur das Handover gestartet wird. Es kann zudem definiert werden, dass vor dem Starten der Aufgabe den Objekten bestimme Eigenschaften, welche eine Bedingung für das Starten des Handovers sind, zugewiesen werden. Dies ist besonders nützlich in der Simulation, in welcher Objekte über ein Skript in die simulierte Welt eingefügt werden. Beispielsweise kann so die Flag der `Empfangsbereitschaft` des Menschen gesetzt werden.

Abstand Das Vorgehen ist bereits im Task-Inference Modul des Roboters vorhanden. Dieses ist dafür zuständig, die nächste am wahrscheinlichsten erscheinende Aufgabe zu starten. Wenn sich der EE dem Menschen somit auf eine bestimmte Distanz annähert und alle in der PDDL definierten Bedingungen erfüllt sind, dann wird diese Aufgabe automatisch gestartet. Für die Ausführung des R2H Handover muss der Roboter ein Objekt gegriffen haben, der Mensch empfangsbereit sein und der EE muss eine bestimmte Distanz zum Menschen unterschritten haben.

Vorbedingung für das Handover

Die Eigenschaft der Empfangsbereitschaft wird im Gelenkpositionsbestimmungsprozess implementiert. Einem detektierten Menschen soll die Eigenschaft empfangsbereit zugewiesen werden, wenn dieser seine rechte Hand ausstreckt.



(a) Empfangsbereiter Mensch (b) Nicht empfangsbereiter Mensch

Abbildung 4.6: Veranschaulichung der Geometrie zur Bestimmung der Empfangsbereitschaft

Die Graphik 4.6 veranschaulicht anhand von zwei Zeichnungen die geometrischen Definitionen für die Berechnung der Empfangsbereitschaft. Um die Empfangsbereitschaft zu bestimmen, werden zwei 3D Vektoren mittels der erkannten Gelenke gebildet. Beide Vektoren gehen vom Schultergelenk (Gelenkpunkt 2) aus. Der eine Vektor endet in der Hand (Gelenkpunkt 4), in der Zeichnung 4.6 in grün dargestellt. Der andere Vektor endet in der rechten Hüfte (Gelenkpunkt 8), hier in gelb dargestellt. Um die Empfangsbereitschaft aus diesen beiden 3D Vektoren zu generieren, wird der Winkel zwischen diesen beiden bestimmt. In der Zeichnung 4.6 wird der Winkel mit dem Symbol α in der Farbe orange dargestellt. Mathematisch dargestellt ergibt sich die Formel 4.7:

$$\alpha = \arccos\left(\frac{\overrightarrow{G2G4} \cdot \overrightarrow{G2G8}}{|\overrightarrow{G2G4}| \cdot |\overrightarrow{G2G8}|}\right) \tag{4.7}$$

mit

$$\begin{aligned} G2 &= \text{Gelenkpunkt 2 - rechte Schulter} \\ G4 &= \text{Gelenkpunkt 4 - rechte Hand} \\ G8 &= \text{Gelenkpunkt 8 - rechte Hüfte} \\ \alpha &= \text{Winkel zwischen den Vektoren} \end{aligned}$$

Um die ideale Grenze zur Bestimmung von α zu finden, werden mehrere Versuche durchgeführt, in welchen der Winkel ausgegeben wird. Nach diesen Tests wird die Winkelgrenze auf ca. $57,3°$ ($1Rad$) gesetzt. Ist der Winkel größer, so wird die Eigenschaft der Empfangsbereitschaft auf `wahr` gesetzt, dies veranschaulicht die Untergraphik 4.6a. Ist der Winkel kleiner, so wird sie auf `falsch` gesetzt, dies zeigt die Untergraphik 4.6b. Es wurde sich für diese Auslöser entschieden, da für diese bereits Grundstrukturen im System (sowohl Softwareseitig als auch Hardwareseitig) vorhanden sind und es das Ziel dieser Arbeit ist, das beidseitige Handover am aktuellen System mit den derzeit vorhandenen Möglichkeiten umzusetzen.

4.2.3 Festlegung und Implementierung der Bewegungsabläufe, welche nach den Auslösern ausgeführt werden sollen

Das Ziel ist es, drei verschiedenen Versionen des R2H Handover zu erstellen. Die Handover Varianten steigern sich dabei in ihrer Komplexität und Unterstützungfunktion während der Ausführung.

Handover Version 1

Die YAML-Datei zu dieser Handover Variante ist im Anhang unter A zu finden. Die folgende Graphik 4.7 zeigt den Automaten, welcher für das R2H definiert wurde.

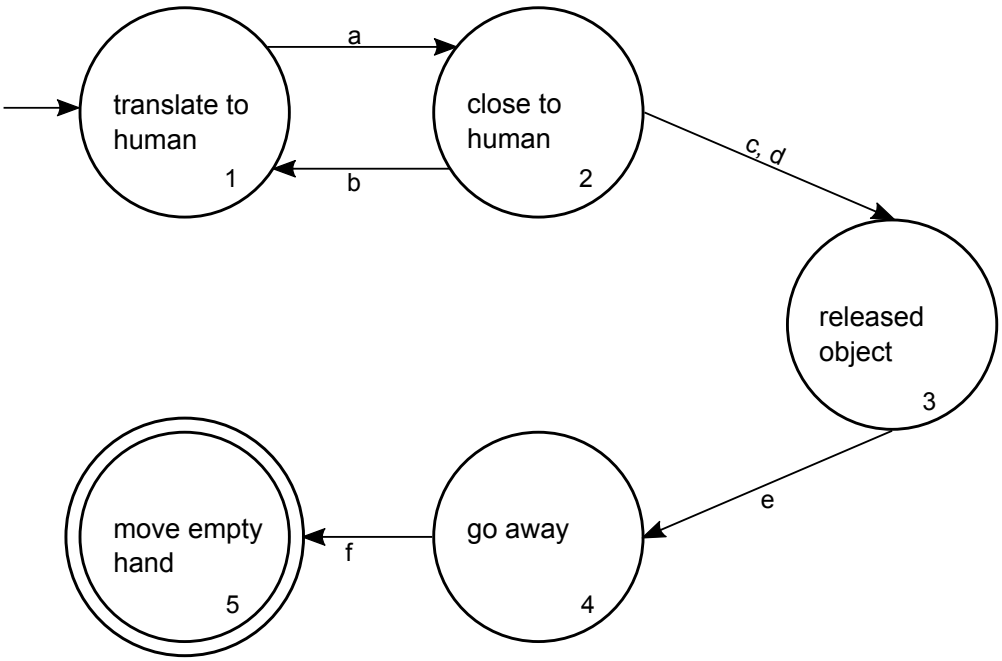


Abbildung 4.7: Automat der ersten Version des R2H Handover

Initiale Definitionen

- Für das Handover werden die Objekte Mensch, Container (beinhaltet Flaschen, Becher usw.) und ein Manipulator benötigt. Dies wird im Listing A.1 in Zeile 1 definiert.
- Vorbedingung: Der Container (das Objekt, welches Überreicht werden soll) muss an den Roboterarm gebunden sein. Der Mensch muss empfangsbereit sein. Zeile 2 in Listing A.1 beschreibt dies.

- Die Zuweisung von `target, manipulator, actor`. Dies beschreiben die Zeilen 3 und 4 in Listing A.1.

Zustände und deren Bedingungen
(Ab Zeile 10 in Listing A.1).

1. **`translate to human`** Das Objekt darf nicht rotiert werden. Es soll senkrecht vom Roboter gehalten werden. Die Bewegung des Roboters soll verlangsamt werden.

2. **`close to human`** Die Bewegung wird nochmals verlangsamt.

3. **`release object`** Die Hand wird geöffnet.

4. **`go away`** Der Effekt des `release object` Zustandes ist hier wie in Listing 4.4 umgesetzt. Entfernen entlang der negativen Z-Achse, ausgehend vom Koordinatensystem des EE. Die Benutzereingaben X und Y werden auf diese angewendet. In der Z-Achse des Weltkoordinatensystems wird lediglich die Bewegungsgeschwindigkeit begrenzt.

5. **`move empty hand`** Keine Bedingungen. Erfolgreicher Zielzustand des Automaten.

Übergänge und deren Bedingungen

a Die Distanz des EE zur Hand des Menschen ist kleiner als 0.25 Meter. (Listing A.1 Zeile 15-22)

b Die Distanz des EE zur Hand des Menschen ist größer als 0.26 Meter.(Listing A.1 Zeile 47-45)

c Krafteinwirkung entlang der X-Achse (im Weltkoordinatensystem) auf den Roboterarm ist größer als 20 Newton.(Listing A.1 Zeile 33-39)

d Krafteinwirkung entlang der Y-Achse (im Weltkoordinatensystem) auf den Roboterarm ist größer als 8 Newton.(Listing A.1 Zeile 40-46)

e Timeout nach 0.2 Sekunden.(Listing A.1 Zeile 65-68)

f Die Distanz zum Menschen ist größer als 0.3 Meter.(Listing A.1 Zeile 76-83)

Zusammengefasst wird nach dem Starten des Handovers im ersten Zustand `translate to human` das Hinbewegen zum Menschen ermöglicht. Hierbei wird die Bewegung direkt übertragen und ist an keine Achsen gebunden. Wird dabei die Distanz von 0.25 Metern unterschritten, so wird in den Zustand `close to human` gewechselt. Hierbei wird die Geschwindigkeit nochmals reduziert. In diesem Zustand befindet sich der EE schon sehr nahe am Menschen, es sollen nur noch kleine Positionsanpassungen vorgenommen werden. Der Übergang in den Zustand `release object` erfolgt mittels der Bestimmung des Drehmoments, welches auf den Roboter einwirkt. Hierbei wird die Kraft in der X und Y Achse betrachtet. Wird in einer Achse die Kraft überschritten, so wird der Zustand gewechselt. Im `release object` Zustand wird die Hand geöffnet. Nach diesem Zustand wird im `go away` Zustand darauf geachtet, dass die Roboterhand sich kontrolliert aus dem Arbeitsbereich des Menschen entfernen kann. Zudem wird in diesem Zustand der Effekt des frei werden der Roboterhand sowie das Auflösen der Bindung zwischen Roboterhand und Objekt ausgelöst. Dies wird mithilfe eines IM realisiert. Diese wird erläutert und dargestellt in Listing 4.1. Ist diese Distanz überwunden, so wird nach einem Timeout im Zustand `move empty hand` das Handover beendet. Dies ist der Zielzustand des Automaten, wird dieser erreicht, wird die Handover Aufgabe beendet. Die Information steht daraufhin allen anderen Prozessen des Systems zur Verfügung. Derer Gelenkpositionserkennungsprozess erhält ebenfalls die Information und beginnt daraufhin damit die Position des Menschen wieder laufend zu aktualisieren.

Handover Version 2

Die zweite Variante des Handovers soll die Übergabe des Objektes erleichtern. Dies wird durch das Hinzufügen einer zusätzlichen Bedingung während des ersten Zustande `translate to human` realisiert.

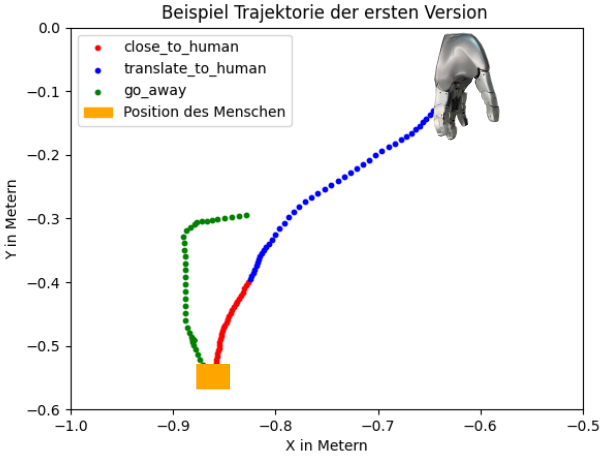
Im Unterschied zu der ersten Variante ist die Hand des Roboters im Zustand `translate to human` zum Menschen hin ausgerichtet. Damit wird verhindert, dass es Verzögerungen im Handover durch eine falsche Position der Hand des Roboters gibt. Realisiert wird dies mittels einer AC, das Listing dieser ist unter 4.2 zu finden.

Handover Version 3

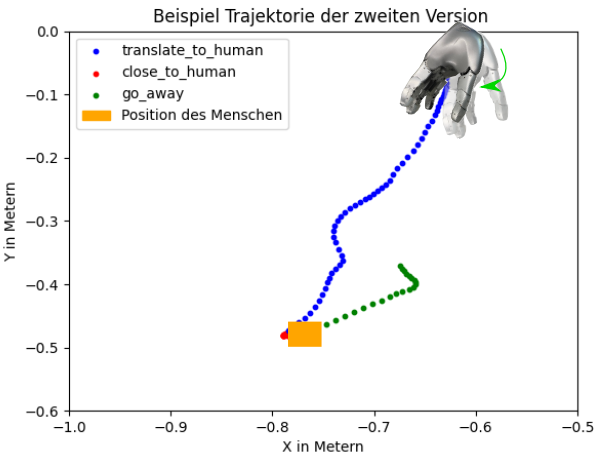
Die dritte Version baut auf die im vorherigen Unterkapitel beschriebene Version zwei des R2H Handover auf. In dieser Version soll eine zusätzliche Erleichterung für den Benutzer des Systems geschaffen werden. Hierfür wird ein weiteres IM dem ersten Zustand des Automaten unter 4.7 hinzugefügt. Dieses beschränkt die Bewegungsfeiheit des Armes, so dass sich der EE lediglich auf einem Vektor, aufgespannt zwischen dem EE und dem Menschen, bewegen kann. Das IM wird so definiert, dass die Eingaben in der X- und Y-Achse mittels eine Skalarproduktes auf diesen Vektor abgebildet werden. Die Achsen werden ausgewählt, um das Handover für den Benutzer möglichst intuitiv zu gestalten. Die restlichen Zustände werden aus dem Automaten der ersten Version übernommen.

Vergleich der Handover Versionen anhand ihrer Tajektorien

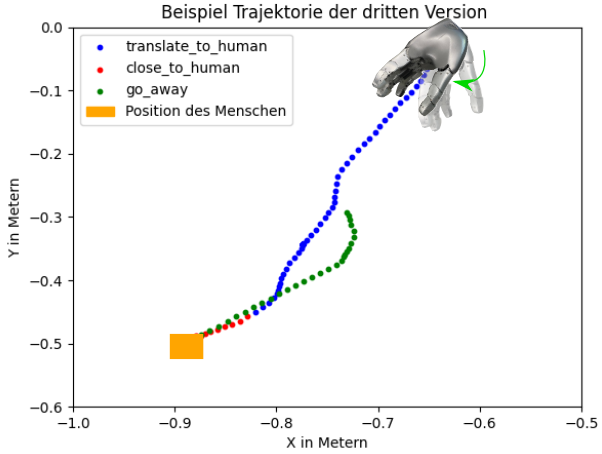
Die Graphiken 4.8 zeigen exemplarisch jeweils eine Trajektorie pro Handover Version. Die Trajektorien wurden während der Durchführung Nutzerstudie aufgezeichnet. Die große Hand zu Beginn der Graphik symbolisiert den Startpunkt der Aufgabenausführung. Die dargestellte Handposition beschreibt die Ausrichtung der Hand. In Version 1 ist zu erkennen, dass sich die Hand während des gesamten Handovers nicht verändert. Bei den anderen beiden Versionen wird die Roboterhand hin zur menschlichen Hand ausgerichtet. Zudem ist in den Graphiken zu erkennen, dass Version 1 und Version 2 keine geometrischen Zwangsbedingungen in der ersten Phase besitzen. In der dritten Version ist dies direkt zu Beginn zu erkennen.



(a)



(b)



(c)

Abbildung 4.8: Trajektorien der Handover Varianten

4.2.4 Einarbeitung in ein Modul zur Automatisierung von SCT um das Handover zu automatisieren

Das Automatisierungsmodul verwendet einen Plan, welcher die Reihenfolge der auszuführenden Zustände zum Erreichen des Zielzustandes beschreibt [26]. Das Listing 4.5 zeigt einen solchen Plan.

Listing 4.5: Beispiel eines Automatisierungsplans

```
1 {translate_to_human: close_to_human,
    close_to_human: released_object, released_object:
        go_away, go_away: move_empty_hand}
```

Die Informationen über die einzelnen Zustände erhält das Modul über die definierten SCT Skills.

Das Automatikmodul kann in jedem Zustand durch einen Auslöser gestartet werden. Dies ermöglicht es, eine gestartete Aufgabe von dem Automatikmodul beenden zu lassen [26].

4.2.5 Testen der Handover in der Simulation

Die drei Handover-Varianten wurden im ersten Schritt in der Simulation entwickelt und getestet. Hierfür wird über einen Simulationsprozess ein Mensch und ein Objekt, welches überreicht werden soll, zur simulierten Welt hinzugefügt. Wie am echten System kann der Roboterarm in der Simulation mittels einer Space Mouse kommandiert werden. Zum Starten des Handovers ist es in der Simulation am einfachsten, ein Button in der bereits ins System integrierten GUI zu definieren. Theoretisch müsste im ersten Schritt das Objekt vom Roboter gegriffen werden, sodass dieser es in seiner Hand hält. Aufgrund des hohen zeitlichen Aufwandes, dass vor jedem Testdurchlauf ein Objekt gegriffen werden muss, wird in der GUI vor dem Starten des Handovers der initial Zustand gesetzt.

Der ideale Kraftwert, welcher überschritten wird, damit sich die Hand des Roboters öffnet, kann in der Simulation nicht herausgefunden werden. Es ist aber möglich, die

Krafteinwirkung in einem Simulationsprozess zu umgehen und die Krafteinwirkung durch einen Auslöser in der Simulation auszulösen. Dies ermöglicht es, dass gesamte Handover ohne die Krafteinwirkung zu testen.

4.2.6 Testen des Handovers am realen System

Am realen System werden für echte Tests mehrere Personen benötigt. Da dies nicht immer möglich ist, kann auch hier die Funktion zum Hinzufügen von Objekten verwendet werden. Damit wird dem Robotersystem simuliert, dass sich ein Mensch an einer bestimmten Position befindet (auch wenn dieser nicht wirklich dort ist). Um das Handover am realen System zu starten, kann ebenfalls die zuvor beschriebene GUI verwendet werden. Es ist aber auch möglich, das Handover von der Task- Inference starten zu lassen, indem sich der Roboter dem Menschen auf eine gewisse Distanz annähert. Während diesen Tests kann viel ausprobiert werden, was die exakten Parameter während des Handovers betrifft. Wird das Handover in Kombination mit der Gelenkpositionsbestimmung getestet, so wird eine zweite Person benötigt. Da nicht immer eine zweite Person zur Verfügung steht, wird ein menschenähnlicher Dummy verwendet. Der Aufbau bei einem Versuch mit dem Dummy ist auf dem Foto 4.9 zu erkennen.

Abbildung 4.9: Testaufbau bei der Verwendung des Dummys

Eine ausführliche Analyse des Handovers ist in Kapitel 4.5 zu finden.

4.3 Integration einer neuen Objekterkennung, um verdeckte Objekte besser verfolgen zu können

Bevor die neue Objekterkennung integriert wird, soll überprüft werden, ob diese überhaupt benötigt wird. Es werden Tests durchgeführt, welche aussagen, wie gut die aktuell verwendete Objekterkennung die zu überreichenden Objekte in der Hand lokalisiert. Bei diesen Tests werden die tatsächlichen Position des Objektes mit den detektierten verglichen.

Dabei wird festgestellt, dass Objekte akkurat von der aktuellen Objekterkennung erkannt werden, während diese von einem Menschen gegriffen werden. Es muss lediglich darauf geachtet werden, dass Objekt nicht vollständig zu verdecken.

Daher ist es nicht erforderlich, eine neue Objekterkennung zu integrieren.

4.4 Implementierung und Testen des Human to Robot Handover (H2R)

Für die Implementierung des H2R Handover wird, wie auch für das R2H Handover, ein Skill im Stil der SCT angelegt. Die Grundlagen sowie der Aufbau eines solchen Skills wurden bereits in Kapitel 4.2.1 erklärt.

4.4.1 Recherche sowie Auswahl verschiedener Auslöser, welche für das Handover verwendet werden können

Für diese Handover werden ebenfalls Auslöser benötigt. Zum Starten des Handovers und für den Zustandswechsel zwischen den Zuständen des Automaten wird ein solcher Auslöser gebraucht. Grundsätzlich sind hier dieselben Auslöser wie für das R2H (Kapitel 4.2.2) möglich. Es wurde sich für dieselben Auslöser, wie für die andere Handoverrichtung, entschieden.

4.4.2 Definition des H2R

Die Graphik 4.10 stellt den Automaten, welcher das H2R Handover definiert, dar.

Initiale Definitionen

- Für das Handover werden die Objekte Mensch, Container (beinhaltet Flaschen, Becher usw.) und ein Manipulator benötigt.
- Vorbedingungen: Der Roboter hat nichts in der Hand. Der Mensch muss bereit sein.
- Zuweisung von `target, manipulator, actor`.

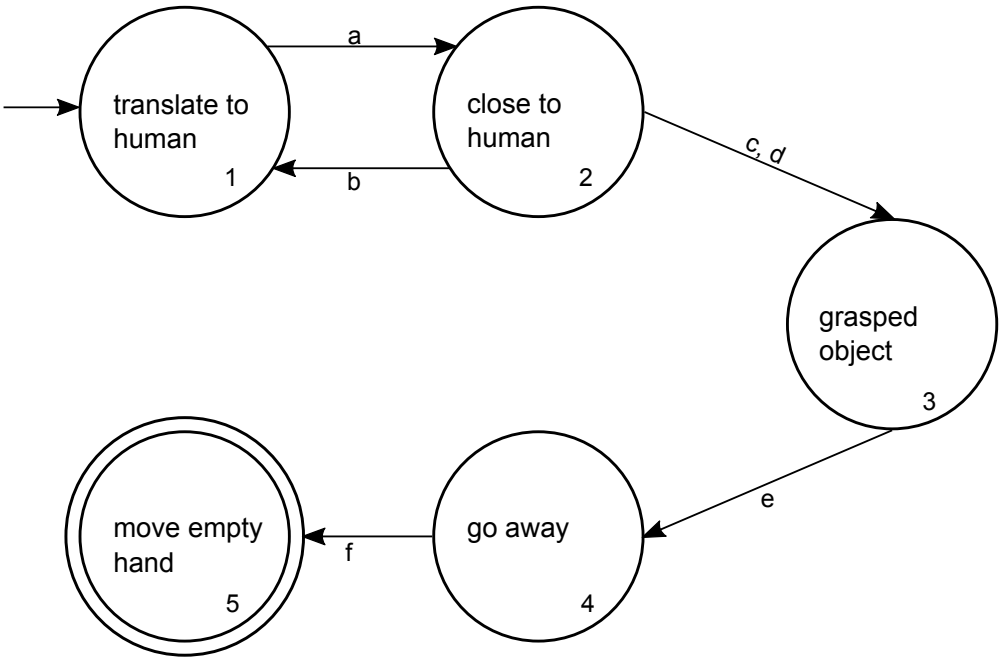


Abbildung 4.10: Automat des H2R Handover

Zustände und deren Bedingungen

1. `translate to human / object` Während sich der Roboter auf den Menschen zubewegt, soll seine Hand aufrecht sein.

2. `close to human` Die Bewegung des Roboters wird nochmals verlangsamt.

3. `grasp` Das Objekt wird gegriffen / die Hand schließt sich.

4. `go away` Der Effekt des Zustandes `grasp` wird hier umgesetzt. Das Objekt ist daraufhin an den Roboterarm gebunden und dieser ist nicht mehr frei. Entfernen entlang der negativen Z-Achse, ausgehend vom Koordinatensystem des EE. Die Benutzereingaben X und Y werden auf diese angewendet. In der Z-Achse des Weltkoordinatensystem wird lediglich die Bewegungsgeschwindigkeit begrenzt.

5. `move empty hand` Keine Bedingungen. Erfolgreicher Zielzustand des Automaten.

Übergänge und deren Bedingungen

a Die Distanz des EE zur Hand des Menschen ist kleiner als 0.25 Meter.

b Die Distanz des EE zur Hand des Menschen ist größer als 0.26 Meter.

c Krafteinwirkung entlang der X-Achse (im Weltkoordinatensystem) auf den Roboterarm ist größer als 20 Newton.

d Krafteinwirkung entlang der Y-Achse (im Weltkoordinatensystem) auf den Roboterarm ist größer als 8 Newton.

e Timeout nach 0.2 Sekunden.

f Die Distanz zum Menschen ist größer als 0.3 Meter.

Die H2R Versionen sind gleich aufgebaut wie die R2H Handover Varianten. Es gibt ebenfalls drei Varianten mit unterschiedlicher Komplexitäts und Unterstützungsstufe. Für die Varianten zwei und drei wurden dieselben Änderungen im ersten Zustand des Automaten vorgenommen, wie beim R2H Handover. Der Grundautomat des R2H und der des H2R unterscheiden sich lediglich in einem Zustand. Anstatt des `release object` Zustandes gibt es im H2R Handover den `grasped object` Zustand. In diesem wird die Hand des Roboters geschlossen, das Objekt, welches gegriffen wurde, wird als `an den Roboter gebunden` und der Roboter als nicht frei definiert.

4.4.3 Integration des vollständigen Handovers am System sowie die Durchführung von Tests

Um das Handover zu entwickeln und zu testen, wird im ersten Schritt der Weg über die Simulation gewählt. Nachdem dieser Schritt erfolgreich war, wird das Handover am realen System getestet. Hierbei wird bei den ersten Versuchen ebenfalls zuerst der Mensch über einen Simulationsprozess und nicht über den Gelenkpositionsbestimmungsprozess hinzugefügt. Die genaue Analyse und Evaluierung der Tests des Handovers sind im nächsten Kapitel 4.5 zusammengefasst.

4.5 Durchführung und Evaluierung einer Nutzerstudie

Zur Evaluierung der implementierten Handover Varianten wird eine kleine Nutzerstudie durchgeführt. Im ersten Schritt werden die Kriterien für die Nutzerstudie aufgestellt. Hierfür wird eine Literaturrecherche durchgeführt. Im zweiten Schritt wird die Studie in einem zuvor definierten Versuchsaufbau durchgeführt. Der letzte Schritt umfasst die Auswertung und Evaluierung der Nutzerstudie. Das Ziel der Studie ist es, herauszufinden, welche der Handover Varianten der Benutzer des Systems präferiert. Zudem soll überprüft werden, ob es Anwendungsunterschiede in den einzelnen Varianten gibt. Da sich die Varianten der beiden Handoverrichtungen bis auf einen Zustand ähneln, wird nur eine Handoverrichtung untersucht. Dies ist ausreichend, da die Unterschiede der verschiedenen Varianten in beiden Richtungen dieselben sind.

4.5.1 Recherche zur Bestimmung von Kriterien für die Nutzerstudie

Eine grundlegende Recherche wurde bereits in der T3_3000 Hausarbeit vorgenommen [1]. Mögliche Kriterien lassen sich in drei verschiedene Kategorien zusammenfassen[1][2]:

- Performance Metriken
- Psychophysiologische Metriken
- Subjektive Metriken

In dieser Studie sollen sowohl Performance Metriken als auch Subjektive Metriken angewendet werden. Auf Psychophysiologische Metriken wird verzichtet, da diese nicht benötigt werden, um die aufgestellten Thesen zu untersuchen. Hierfür reichen die anderen beiden Metriken aus.

Zur Bestimmung der Performance Metriken werden Daten am System während der Nutzerstudie aufgezeichnet. Aus diesen werden Durchschnittswerte für die einzelnen Varianten gebildet. Es wird die durchschnittliche Zeit für die Ausführung

der jeweiligen Handover Variante bestimmt so wie die Erfolgsrate. Des Weiteren wird versucht, aus den gesammelten Daten Rückschlüsse auf die Zuverlässigkeit des Handovers zu ziehen [27]. Zudem werden die Trajektorien des EE während des Handovers aufgezeichnet.

Mittels subjektiver Metriken sollen Aussagen über das Empfinden der Probanden während der Ausführung des Handovers getroffen werden. Ein Kriterium, welches untersucht werden soll, ist das Vertrauen in die Methode [28]. Das subjektive Empfinden der Probanden wird mit einem Fragenkatalog erfasst.

Es werden einige Fragen aus dem Godspeed Test verwendet [29]. Aus diesen Fragen kann geschlossen werden, wie sich das Handover für den Probanden anfühlt. Es werden gezielte Fragen gestellt zur Feststellung wie natürlich / künstlich das Handover von den Probanden empfunden wurde [30].

Zudem werden Fragen aus dem **N**ational **A**eronautics and **S**pace **A**dministration (NASA) **T**ask **L**oad **I**ndex (TLX) Test verwendet [31]. Diese geben Aufschluss darüber, wie anstrengend der jeweilige Proband das Handover empfunden hat.

Der Fragenkatalog ist in zwei Teile aufgeteilt, basierend darauf zu welchem der beiden Tests die Fragen gehören. Zu Beginn werden Fragen aus dem NASA TLX gestellt. Es werden jeweils Fragen zu verschieden Arten von Belastungen gestellt und mittels einer Zahlen Skala wird vom Probanden angegeben, wie stark die jeweilige Belastung war. Auf der Skala kann von null bis fünf geantwortet werden. Null entspricht einer sehr geringen Belastung und fünf einer sehr hohen. Ziel dieser geraden Skala ist es, dass kein Mittelpunkt getroffen werden kann und sich für eine Richtung entschieden werden muss. Dies sind die verwendeten Fragen [31] :

1. How mentally demanding was the task?
2. How physically demanding was the task?
3. How hard did you have to work to accomplish your level of performance?
4. How insecure, discouraged, irritated, stressed and annoyed were you?

Für den zweiten Teil wird die gleiche Skala verwendet. In diesem Teil sind jeweils zwei gegensätzliche Begriffe gegeben, basierend auf dem Godspeed Test [29]. Mittels der

Skala wird angegeben, welchem Begriff zugestimmt wird. Die folgenden Begriffspaare werden verwendet:

5. fake - natural
6. artificial - lifelike
7. moving rigidly - moving elegantly
8. unpleasant - pleasant
9. anxious - relaxed

Im Anhang unter B ist der erstellt Fragenkatalog beigefügt.

4.5.2 Durchführung der Nutzerstudie und Anwendung der oben genannten Kriterien

Die Nutzerstudie wird mit fünf Probanden durchgeführt. Vier der Probanden haben alle bereits Erfahrung im Umgang mit der Steuerung des Roboters. Die Probanden führen das Handover in der Rolle des Benutzers aus. Dies bedeutet, dass sie den Roboter steuern. Es wird die Handoverrichtung R2H verwendet. Es werden alle drei in Kapitel 4.2.3 beschriebenen Varianten getestet. Diese werden nacheinander durchgeführt. Dabei wird mit Handover Variante eins gestartet und mit Variante drei geendet. Jeder Proband befindet sich zur Durchführung an der gleichen Stelle und hat damit dasselbe Sichtfeld. Die Person, welche den Gegenstand in Empfang nimmt, befindet sich ebenfalls bei jedem Durchgang an einer ähnlichen Position.

(a) (b) (c) (d)

Abbildung 4.11: Bilderserie der Nutzerstudie

Die Bilderserie 4.11 zeigt einen Durchlauf der Handover Version 2 in der Nutzerstudie. Bild (a) spiegelt den Initial-zustand vor dem Beginn des Handovers wider. Auf Bild (b) ist zu sehen, dass die Hand des Roboters durch den Start der Handover Aufgabe im Vergleich zum Bild davor zum Menschen gedreht ist. Der Kontakt mit dem Menschen und das damit verbundene Öffnen der Hand ist auf Bild (c) sichtbar. Das letzte Bild (d) zeigt den Zustand `go away`.

4.5.3 Auswertung der Nutzerstudie

Grundsätzlich ist es schwierig, vergleichbare Studien zu finden, da es wenige bis keine Veröffentlichungen zu Handoverszenarien gibt, bei welchen es einen Benutzer wie an diesem Robotersystem gibt. Die Hauptzahl der Robotersysteme, an welchen Handoverszenarien durchgeführt wurden, sind darauf ausgelegt, die Aufgabe autonom durchzuführen, ohne dass sie von einem Nutzer gesteuert werden. Der besondere Unterschied in dieser Implementierung ist es, dass die Nutzerstudie aus der Sicht des Benutzers des System durchgeführt wird und nicht aus der Sicht der Person die den Gegenstand vom Roboter empfängt. In den folgenden zwei Unterkapiteln werden die Ergebnisse der Studie ausgewertet.

Auswertung nach Performance Metriken

Zur Bestimmung der Performance der Handover Varianten wurden die Zeiten für die einzelnen Phasen des Handover gemessen. Die durchschnittlich gemessenen Zeiten während der einzelnen Zustände sind in der Graphik 4.12 visualisiert.

Dargestellt werden die drei Phasen `translate to human`, `close to human` und `go away`, die anderen Phasen des Handovers werden nicht dargestellt, da diese als Übergangsbedingung eine bestimmte Zeitspanne definiert haben und diese daher bei allen drei Varianten gleich sind. In der Graphik 4.12 ist zu erkennen, dass im ersten Zustand die Zeit mit jeder Variante steigt, in den anderen Phasen ist dies unterschiedlich. Die Zeitdarstellung in der Phase `go away` lässt vermuten, dass dies den Probanden, da Variante 2 und 3 in diesem Zustand gleich sind, den Benutzern schon bekannt war und die Zeitersparnis daher kommt. Die durchschnittliche Zeit

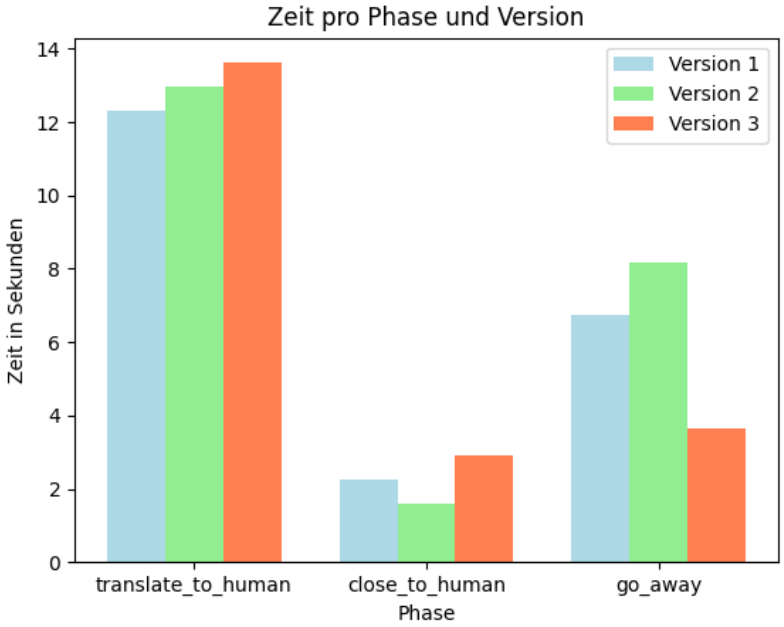


Abbildung 4.12: Durchschnittliche Dauer der Handoverphasen

der kompletten ersten Variante liegt bei 22.2 Sekunden. Die zweite Variante besitzt die Durchschnittszeit 23.4 Sekunden und ist damit die längste Durchschnittszeit. Mit 20.8 Sekunden hat die dritte Variante die kürzeste Durchschnittszeit.

Von insgesamt 15 Handover Durchführungen (alle drei Varianten) waren 14 erfolgreich. Dies spricht dafür, dass die Handover Varianten zuverlässig sind. Version 1 und Version 3 erreichen eine Erfolgsrate von 100%. Die Erfolgsrate von Variante 2 liegt bei 93.33%, da in dieser Variante sich die Hand des Roboters einmal zu früh geöffnet hat. Der Fehler lag an der Position des Empfängers, die Hand des Empfängers befand sich an einer ungünstigen Position. Zum Erreichen der Position der menschlichen Hand musste der Arm des Roboters gestreckt werden, dies führte dazu, dass ohne die Krafteinwirkung des Menschen bereits die benötigte Kraft gemessen wurde, weshalb sich die Roboterhand zu früh geöffnet hatte.

Auswertung nach subjektiven Metriken

Die subjektive Wahrnehmung der verschiedenen Handover Varianten wird mittels des Fragebogens unter B ermittelt. Das Ergebnis ist in zwei Teile aufgeteilt. Der erste Teil beinhaltet die ersten vier Fragen, diese stammen aus dem NASA TLX Test. Die Graphik 4.13 visualisiert die Ergebnisse.

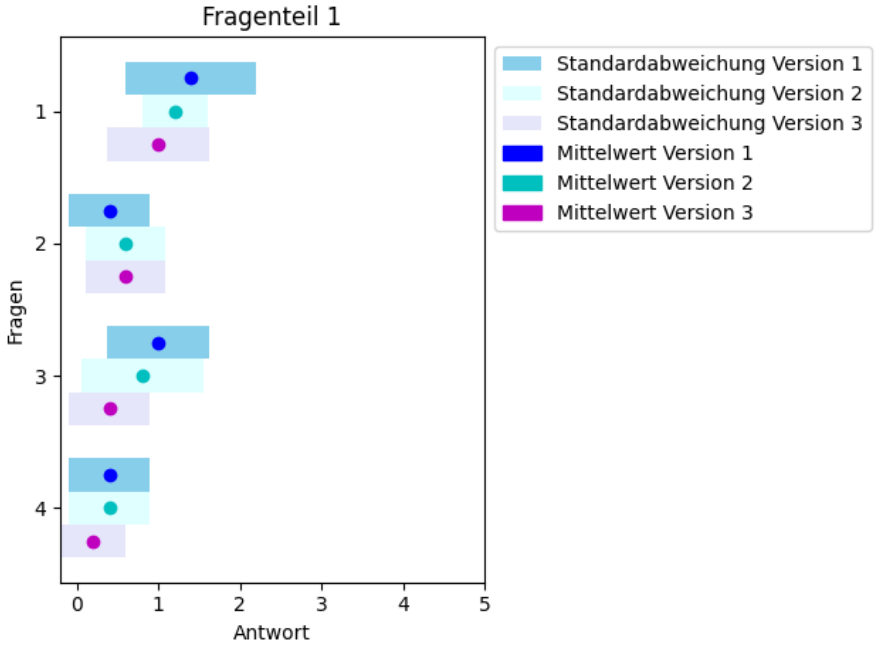


Abbildung 4.13: Ergebnisse des Fragebogens: Teil 1

Aus der Graphik 4.13 ist zu schließen, dass die Ausführung des Handovers (in allen drei Versionen) für den Benutzer nicht viel Anstrengung erfordert. Anhand der Auswertung ist zu erkennen, dass die erste und zweite Version ähnlich bewertet wurden. Die dritte Version wurde in den Fragen eins, drei und vier als weniger anstrengend bewertet.

Der zweite Teil der Auswertung des Fragebogens ist in der Graphik 4.14 veranschaulicht.

In der Graphik 4.14 wird deutlich, dass im Schnitt alle Handover Varianten als natürlich und angenehm von den Probanden empfunden wurden. Auffallend ist, dass Version 2 bei allen fünf Fragen den höchsten Durchschnittswert erreicht hat und

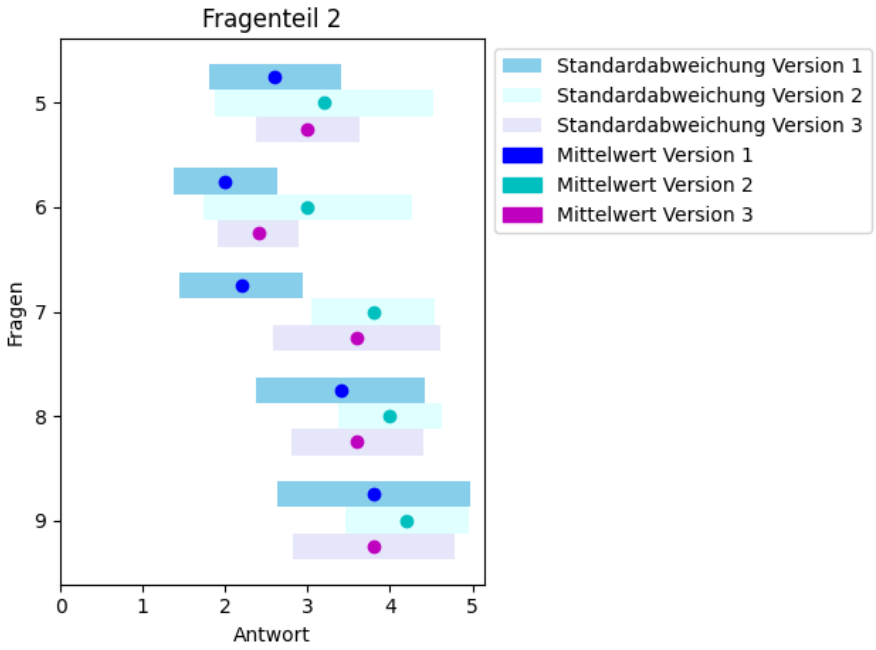


Abbildung 4.14: Ergebnisse des Fragebogens: Teil 2

gleichzeitig die Standardabweichung bei diesem bei den Fragen fünf und sechs am höchsten ist. Dies spricht dafür, dass diese Variante im Durchschnitt von den Nutzern am wenigsten künstlich empfunden wird. Jedoch aber gleichzeitig mit Blick auf die Standardabweichung die konkreten Meinungen auseinander driften. Variante 1 wurde im Durchschnitt als am unnatürlichsten empfunden, dies ist daran zu erkennen, dass der Durchschnitt am niedrigsten ist.

Aus beiden Umfrageteilen kombiniert lässt sich schließen, dass die zweite Variante am meisten Zeit benötigt hat, aber von den Nutzern als am natürlichsten und angenehmsten empfunden wurde. Die dritte Variante war am einfachsten auszuführen für die Probanden.

5 Ergebnis

Im Folgenden wird in Tabellenform das Ergebnis der Arbeit dargestellt.

Nummer	Aufgabe	Status
1	**Bestimmung der physischen Position des Menschen und Verarbeitung dieser Position für das Handover**	
1.1	Literaturrecherche zur Positionsbestimmung des Menschen im Hinblick auf die Auswahl eines geeigneten Verfahrens für die Handover	✓
1.2	Implementierung und Integration einer Methode zur Gelenkpositionsbestimmung des Menschen	✓
1.3	Evaluierung der Möglichkeiten zur Bestimmung einer 3D Position des Menschen	✓
1.4	Integration des Prozesses ins reale System sowie das Testen des Prozesses	✓

Tabelle 5.1: Ergebnis 1

Nummer	Aufgabe	Status
2	**Implementieren und Testen der Übergabe von Gegenständen / Objekten von Roboter zum Menschen (Robot to Human R2H)**	
2.1	Einarbeitung in Shared Control Templates (SCT)	✓
2.2	Recherche und Auswahl verschiedener Auslöser, welche für das Handover benötigt werden	✓
2.3	Festlegung und Implementierung der Bewegungsabläufe, welche nach den Auslösern ausgeführt werden sollen	✓
2.4	Einarbeitung in ein Modul zur Automatisierung von SCT, um das Handover zu automatisieren	✓
2.5	Integration des vollständigen Handovers am System sowie die Durchführung von Tests	✓

Tabelle 5.2: Ergebnis 2

Nummer	Aufgabe	Status
3	**Integration einer neuen Objekterkennung, um verdeckte Objekte besser verfolgen zu können**	✓

Tabelle 5.3: Ergebnis 3

Nummer	Aufgabe	Status
4	**Implementierung und Testen des Human to Robot Handover (H2R)**	
4.1	Recherche sowie Auswahl verschiedener Auslöser, welche für das Handover verwendet werden können	✓
4.2	Festlegung und Implementierung der Bewegungsabläufe, welche nach den Auslösern ausgeführt werden sollen	✓
4.3	Integration des vollständigen Handovers am System sowie die Durchführung von Tests	✓

Tabelle 5.4: Ergebnis 4

Nummer	Aufgabe	Status
5	**Durchführung und Evaluierung einer Nutzerstudie**	
5.1	Recherche zur Bestimmung von Kriterien für die Nutzerstudie	✓
5.2	Durchführung der Nutzerstudie und Anwendung der oben genannten Kriterien	✓
5.3	Auswertung und Evaluierung der Nutzerstudie	✓

Tabelle 5.5: Ergebnis 5

6 Fazit

In diesem Kapitel wird das Ergebnis der Bachelorabeit kritisch reflektiert. Zudem wird ein Ausblick gegeben.

6.1 Kritische Reflexion

Im vorherigen Kapitel 5 wurde gezeigt, welche der unter Kapitel 2 definierten Aufgabenstellungen umgesetzt werden konnten. In dieser Arbeit wurde erfolgreich mit den am Robotersystem vorhandenen Komponenten ein beidseitiges Handover integriert. Im ersten Schritt wurde erfolgreich ein Skript zur Bestimmung der 3D Gelenkposition des Menschen erstellt. Die Lokalisierung erkennt die Position der Gelenke im konkreten Arbeitsbereich. Bei zunehmender Distanz des Menschen zur Kamera nimmt die Genauigkeit ab. Für das Handover ist die Erkennung jedoch ausreichend. Es wurden für jede Richtung des Handovers drei robuste Varianten erstellt. Diese wurden daraufhin in einer kleinen Nutzerstudie evaluiert. Aus zeitlichen Gründen wurde keine größere Studie mit mehreren Probanden durchgeführt. Dies hätte zu einer repräsentativeren Aussage über die Varianten geführt.

Im Vergleich zu anderen Handovervarianten ist das Besondere an den hier implementierten Versionen, dass sie von einem Benutzer, welcher den Roboter steuert, ausgeführt werden und daher das Hauptaugenmerk auf dessen Empfinden gerichtet wird. Dieses Empfinden wurde erfolgreich in der Nutzerstudie untersucht.

6.2 Ausblick

Eine mögliche Erweiterung wäre es, eine andere Tiefenkamera für die Detektion des Menschen zu verwenden. Dies würde helfen, den Menschen in weiterer Entfernung korrekt zu lokalisieren. Durch diese Veränderung wäre es möglich, die Bewegung des Rollstuhls in das Handover mit einfließen zu lassen. Dies würde bedeuten, dass sich der Roboter im ersten Schritt mittels der Rollstuhlbewegung dem Menschen nähert und ab einer bestimmten Nähe das bereits bekannte Handover ausgeführt wird.

Eine mögliche Erweiterung wäre es zudem, den Prozess der Gelenkpositionsbestimmung weiter zu optimieren. Beispielsweise indem nicht nur ein Mensch im Bild erkannt werden kann, sondern mehrere von welchen dann ein bestimmter nach vordefinierten Kriterien ausgewählt wird.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Bearbeitung dieser Bachelorarbeit unterstützt haben.

Zuerst gebührt mein Dank meinen Betreuern dieser Arbeit, Herrn Vogel vom DLR und Herrn Schultheiß von der **D**uale **H**ochschule **B**aden-**W**ürttemberg (DHBW). Ich bedanke mich für die Unterstützung und die konstruktive Kritik während der Erstellung meiner Arbeit.

Des Weiteren möchte ich mich bei all meinen Kollegen und Kolleginnen des DLR für die hilfreichen Diskussionen und unterstützenden Gespräche bedanken. Zudem bedanke ich mich bei den Probanden für die Teilnahme an der Nutzerstudie.

Literatur

[1] Hanna Rebecca Riesch. *T3_300: Recherche zum aktuellen Stand der Forschung im Bereich der Objektübergabe zwischen Mensch und Roboter*. 05/2022.

[2] Valerio Ortenzi u. a. „Object Handovers: A Review for Robotics“. In: *IEEE Transactions on Robotics* 37.6 (12/2021), S. 1855–1873. ISSN: 1941-0468. DOI: 10.1109/TRO.2021.3075365.

[3] *Welcome to Python.Org*. URL: https://www.python.org/about/. (besucht am 15.08.2022).

[4] Srinivas Rao. *Top 10 Benefits of Using Python*. 09/2020. URL: https://blogs.pqube.in/10-benefits-of-using-python/. (besucht am 15.08.2022).

[5] Jörn Vogel u. a. „EDAN: An EMG-controlled Daily Assistant to Help People With Physical Disabilities“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 10/2020, S. 4183–4190. DOI: 10.1109/IROS45743.2020.9341156.

[6] Daniel Leidner, Christoph Borst und Gerd Hirzinger. „Things Are Made for What They Are: Solving Manipulation Tasks by Using Functional Object Classes“. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 11/2012, S. 429–435. DOI: 10.1109/HUMANOIDS.2012.6651555.

[7] A. Craig. „PDDL-The Planning Domain Defintion Language“. In: (2012). URL: https://www.researchgate.net/profile/Craig-Knoblock/publication/2278933_PDDL_-_The_Planning_Domain_Definition_Language/links/0912f50c0c99385e19000000/PDDL-The-Planning-Domain-Definition-Language.pdf. (besucht am 05.08.2022).

[8] Gabriel Quere u. a. „Shared Control Templates for Assistive Robotics“. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 05/2020, S. 1956–1962. DOI: 10.1109/ICRA40945.2020.9197041.

[9] Maged Iskandar u. a. „Employing Whole-Body Control in Assistive Robotics“. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 11/2019, S. 5643–5650. DOI: 10.1109/IROS40897.2019.8967772.

[10] Hussein Haggag u. a. „Measuring Depth Accuracy in RGBD Cameras“. In: *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*. 12/2013, S. 1–7. DOI: 10.1109/ICSPCS.2013.6723971.

[11] A. Albu-Schäffer u. a. „The DLR Lightweight Robot: Design and Control Concepts for Robots in Human Environments“. In: *Industrial Robot: An International Journal* 34.5 (01/2007). Hrsg. von Clive Loughlin, S. 376–385. ISSN: 0143-991X. DOI: 10.1108/01439910710774386.

[12] H. Liu u. a. „Multisensory Five-Finger Dexterous Hand: The DLR/HIT Hand II“. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 09/2008, S. 3692–3697. DOI: 10.1109/IROS.2008.4650624.

[13] Werner Friedl u. a. „CLASH: Compliant Low Cost Antagonistic Servo Hands“. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, 10/2018, S. 6469–6476. DOI: 10.1109/IROS.2018.8593903.

[14] Werner Friedl und Máximo A. Roa. „CLASH—A Compliant Sensorized Hand for Handling Delicate Objects“. In: *Frontiers in Robotics and AI* 6 (01/2020), S. 138. ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00138.

[15] Young Sang Choi u. a. „Hand It over or Set It down: A User Study of Object Delivery with an Assistive Mobile Manipulator“. In: *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*. 09/2009, S. 736–743. DOI: 10.1109/ROMAN.2009.5326254.

[16] Robin Rasch, Sven Wachsmuth und Matthias König. „A Joint Motion Model for Human-Like Robot-Human Handover“. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. 11/2018, S. 180–187. DOI: 10.1109/HUMANOIDS.2018.8624967.

[17] P. Viola und M. Jones. „Rapid Object Detection Using a Boosted Cascade of Simple Features“. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* Bd. 1. 12/2001, S. I–I. DOI: 10.1109/CVPR.2001.990517.

[18] *OpenCV: Cascade Classifier.* 08/2222. URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. (besucht am 05.08.2022).

[19] CMU-Perceptual-Computing-Lab. *OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation.* 04/2017. URL: https://github.com/CMU-Perceptual-Computing-Lab/openpose. (besucht am 19.08.2022).

[20] Zhe Cao u. a. „OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (01/2021), S. 172–186. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2929257.

[21] Daniil-Osokin. *Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose.* 03/2019. URL: https://github.com/CMU-Perceptual-Computing-Lab/openpose. (besucht am 05.08.2022).

[22] Daniil Osokin. „Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose“. In: *CoRR* abs/1811.12004 (2018). URL: http://arxiv.org/abs/1811.12004.

[23] *OpenCV: Camera Calibration.* 08/2022. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. (besucht am 19.08.2022).

[24] *Open3d.Geometry.PointCloud — Open3D 0.15.1 Documentation.* URL: http://www.open3d.org/docs/release/python_api/open3d.geometry.PointCloud.html?highlight=create%20point%20cloud#open3d.geometry.PointCloud.create_from_depth_image. (besucht am 15.08.2022).

[25] Ansgar Koene u. a. „Experimental Testing of the CogLaboration Prototype System for Fluent Human-Robot Object Handover Interactions“. In: *The 23rd IEEE International Symposium on Robot and Human Interactive Communication.* 08/2014, S. 249–254. DOI: 10.1109/ROMAN.2014.6926261.

[26] Samuel Bustamante u. a. „Toward Seamless Transitions Between Shared Control and Supervised Autonomy in Robotic Assistance“. In: *IEEE Robotics and Automation Letters* 6.2 (04/2021), S. 3833–3840. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3064449.

[27] Robin R. Murphy und Debra Schreckenghost. „Survey of Metrics for Human-Robot Interaction“. In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 03/2013, S. 197–198. DOI: 10.1109/HRI.2013.6483569.

[28] Aaron Steinfeld u. a. „Common Metrics for Human-Robot Interaction“. In: *Proceeding of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction - HRI '06*. Salt Lake City, Utah, USA: ACM Press, 2006, S. 33. ISBN: 978-1-59593-294-5. DOI: 10.1145/1121241.1121249.

[29] Christoph Bartneck. *The Godspeed Questionnaire Series*. 03/2008. URL: https://www.bartneck.de/2008/03/11/the-godspeed-questionnaire-series/. (besucht am 09.08.2022).

[30] Christoph Bartneck u. a. „Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots“. In: *International Journal of Social Robotics* 1.1 (01/2009), S. 71–81. ISSN: 1875-4791, 1875-4805. DOI: 10.1007/s12369-008-0001-3.

[31] *TLX @ NASA Ames - Home*. URL: https://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX_pappen_manual.pdf. (besucht am 20.08.2022).

Anhang A

Listing R2H Variante 1

Listing A.1: R2H Version 1

```
1  parameters: (?s - _container ?t - human ?m -
       _manipulator)
2  precondition: (and (bound ?s ?m) (receive_ready ?t))
3
4  actor: (?s - _container)
5  target: (?t - human)
6
7  automaton_plans:
8  - {translate_to_human: close_to_human,
       close_to_human: released_object, released_object:
       go_away, go_away: move_empty_hand}
9
10 states:
11  translate_to_human:
12   parameters:
13    safety_level: 3
14   transitions:
15    - to: close_to_human
16      predicates:
17       - aux_fct:
18           frame: EE
```

```
19          ref: {origin_frame: target, feature:
              START_Of_STATE, plot: True}
20          function: EuclidianDist
21          mapping : [1,1,1,0,0,0]
22          range: [0.25, inf]
23   input_mapping:
24    - mapping: [.6tx,.6ty,.75tz,0,0,0]
25   active_constraints:
26    - frame: EE
27      mapping: [x,x,x,1.57,-1.57,0]
28
29 close_to_human:
30   parameters:
31    safety_level: 3
32   transitions:
33    - to: released_object
34      predicates:
35       - aux_fct:
36          wrench: EE
37          function: StraightMap
38          mapping : [1,0,0,0,0,0]
39          range: [-inf, 20]
40    - to: released_object
41      predicates:
42       - aux_fct:
43          wrench: EE
44          function: StraightMap
45          mapping : [0,1,0,0,0,0]
46          range: [-inf, 8]
47    - to: translate_to_human
48      predicates:
49       - aux_fct:
50          frame: EE
```

```
51            ref: {origin_frame: target, feature:
                START_Of_STATE, plot: True}
52            function: EuclidianDist
53            mapping : [1,1,1,0,0,0]
54            range: [-inf, 0.26]
55    input_mapping:
56     - mapping: [.4tx,.4ty,.4tz,0,0,0]
57    active_constraints:
58     - frame: EE
59       mapping: [x,x,x,1.57,-1.57,0]
60
61  released_object:
62    parameters:
63     EE_config: {entity: actor, config_name:
          release_grasp}
64     safety_level: 3
65    transitions:
66     - to: go_away
67       predicates:
68        - timeout: 0.2
69
70  go_away:
71    effect: (and (not (bound ?s ?m)) (free ?m))
72    parameters:
73     EE_config: {entity: target, config_name:
          release_grasp}
74     safety_level: 3
75    transitions:
76     - to: move_empty_hand
77       predicates:
78       - aux_fct:
79           frame: EE
80           ref: {origin_frame: target, feature:
                START_Of_STATE, plot: True}
```

```
81          function: EuclidianDist
82          mapping : [1,1,0,0,0,0]
83          range: [-inf, 0.3]
84    input_mapping:
85      - relative: True
86        mapping: [0,0,alpha-,0,0,0]
87        auxiliary_functions:
88         alpha:
89          function: Scalar
90          vector: {origin_frame: EE, axis: Z}
91          partial_input_mapping:
92           mapping: [1.3tx, 1.3ty, 0, 0, 0, 0]
93      - mapping: [0,0,.3tz+,0,0,0]
94
95  move_empty_hand:
96    input_mapping:
97      - mapping: [tx,ty,tz,0,0,0]
98    transitions:
99      - to: exit
100       predicates:
101        - timeout: 0.1
```

Anhang B

Fragebogen

ID:

Human to robot Handover – **version 1**

Answer all question in a rating between 0 to 5. Mark the number you are choosing.

1. How mentally demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

2. How physically demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

3. How hard did you have to work to accomplish your level of performance?

Very low *0* *1* *2* *3* *4* *5* *very high*

4. How insecure, discouraged, irritated, stressed and annoyed were you?

Very low *0* *1* *2* *3* *4* *5* *very high*

Rate between this impressions:

fake *0* *1* *2* *3* *4* *5* *natural*

artificial *0* *1* *2* *3* *4* *5* *lifelike*

moving rigidly *0* *1* *2* *3* *4* *5* *moving elegantly*

unpleasant *0* *1* *2* *3* *4* *5* *pleasant*

anxious *0* *1* *2* *3* *4* *5* *relaxed*

ID:

Human to robot Handover – **version 2**

Answer all question in a rating between 0 to 5. Mark the number you are choosing.

1. How mentally demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

2. How physically demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

3. How hard did you have to work to accomplish your level of performance?

Very low *0* *1* *2* *3* *4* *5* *very high*

4. How insecure, discouraged, irritated, stressed and annoyed were you?

Very low *0* *1* *2* *3* *4* *5* *very high*

Rate between this impressions:

fake *0* *1* *2* *3* *4* *5* *natural*

artificial *0* *1* *2* *3* *4* *5* *lifelike*

moving rigidly *0* *1* *2* *3* *4* *5* *moving elegantly*

unpleasant *0* *1* *2* *3* *4* *5* *pleasant*

anxious *0* *1* *2* *3* *4* *5* *relaxed*

ID:

Human to robot Handover – **version 3**

Answer all question in a rating between 0 to 5. Mark the number you are choosing.

1. How mentally demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

2. How physically demanding was the task?

Very low *0* *1* *2* *3* *4* *5* *very high*

3. How hard did you have to work to accomplish your level of performance?

Very low *0* *1* *2* *3* *4* *5* *very high*

4. How insecure, discouraged, irritated, stressed and annoyed were you?

Very low *0* *1* *2* *3* *4* *5* *very high*

Rate between this impressions:

fake *0* *1* *2* *3* *4* *5* *natural*

artificial *0* *1* *2* *3* *4* *5* *lifelike*

moving rigidly *0* *1* *2* *3* *4* *5* *moving elegantly*

unpleasant *0* *1* *2* *3* *4* *5* *pleasant*

anxious *0* *1* *2* *3* *4* *5* *relaxed*